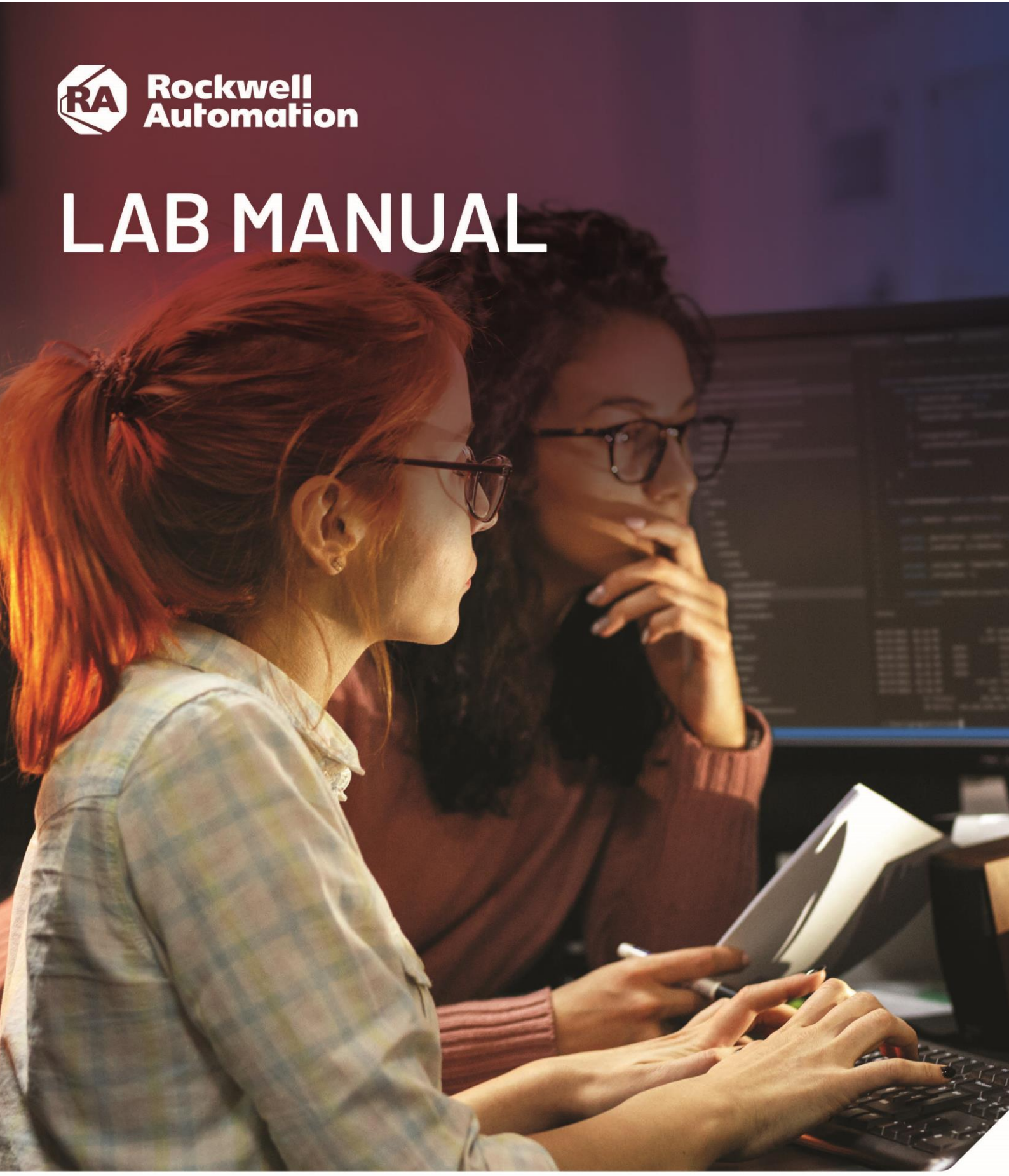
Rockwell Automation

LAB MANUAL



Explore FactoryTalk Optix™

Important User Information

This documentation, whether, illustrative, printed, “online” or electronic (hereinafter “Documentation”) is intended for use only as a learning aid when using Rockwell Automation approved demonstration hardware, software and firmware. The Documentation should only be used as a learning tool by qualified professionals.

The variety of uses for the hardware, software and firmware (hereinafter “Products”) described in this Documentation, mandates that those responsible for the application and use of those Products must satisfy themselves that all necessary steps have been taken to ensure that each application and actual use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards in addition to any applicable technical documents.

In no event will Rockwell Automation, Inc., or any of its affiliate or subsidiary companies (hereinafter “Rockwell Automation”) be responsible or liable for any indirect or consequential damages resulting from the use or application of the Products described in this Documentation. Rockwell Automation does not assume responsibility or liability for damages of any kind based on the alleged use of, or reliance on, this Documentation.

No patent liability is assumed by Rockwell Automation with respect to use of information, circuits, equipment, or software described in the Documentation.

Except as specifically agreed in writing as part of a maintenance or support contract, equipment users are responsible for:

- properly using, calibrating, operating, monitoring and maintaining all Products consistent with all Rockwell Automation or third-party provided instructions, warnings, recommendations and documentation;
- ensuring that only properly trained personnel use, operate and maintain the Products at all times;
- staying informed of all Product updates and alerts and implementing all updates and fixes; and
- all other factors affecting the Products that are outside of the direct control of Rockwell Automation.

Reproduction of the contents of the Documentation, in whole or in part, without written permission of Rockwell Automation is prohibited.

Throughout this manual we use the following notes to make you aware of safety considerations:

WARNING

Identifies information about practices or circumstances
that can cause an explosion in a hazardous environment,
which may lead to personal injury or death, property damage, or economic loss.

IMPORTANT

Identifies information that is critical for successful application and understanding of the product.

ATTENTION

Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you:

- identify a hazard
- avoid a hazard
- recognize the consequence

SHOCK HAZARD

Labels may be located on or inside the drive to alert people that dangerous voltage may be present.

BURN HAZARD

Labels may be located on or inside the drive to alert people that surfaces may be dangerous temperatures.

Explore FactoryTalk Optix™

Contents

Before you begin

This Lab is intended for users who wish to gain an overview of FactoryTalk Optix™, there is no previous experience with FactoryTalk Optix™ required to complete this lab.

About this lab

This Lab is an introductory Lab that explores the features and functionality of FactoryTalk Optix™.

The lab is intended as a complete all sequentially style Lab, with all of the sections intended to be completed one after the other.

There is one Appendix that is optional to further explore the features and functionality of FactoryTalk Optix™.

The lab sections contain an explanation and feature details for that particular section in a “Read about” section. It is not necessary to read these explanations, but a more thorough understanding of the features presented will be attained if time is taken to read through them.

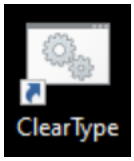
The lab will take approximately 2h30m to complete.

The Lab is based on a process that has an infeed pump, buffer tank, separator, filling machine and bottle capper and is shown below.

Tools & prerequisites

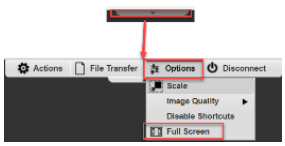
To improve the font clarity within FactoryTalk Optix™ when running in onCourse, we created a script that will enable the Clear Type font.

1. Double click on the ClearType link on the desktop, the script will automatically run and close.



To set the screen to full screen for better viewing of the lab, follow the instruction below.

1. In the middle of the onCourse Window at the top of the screen, you will see a **pull down bar**. When you click on it and go to **Options**, you can select **Full Screen** mode.



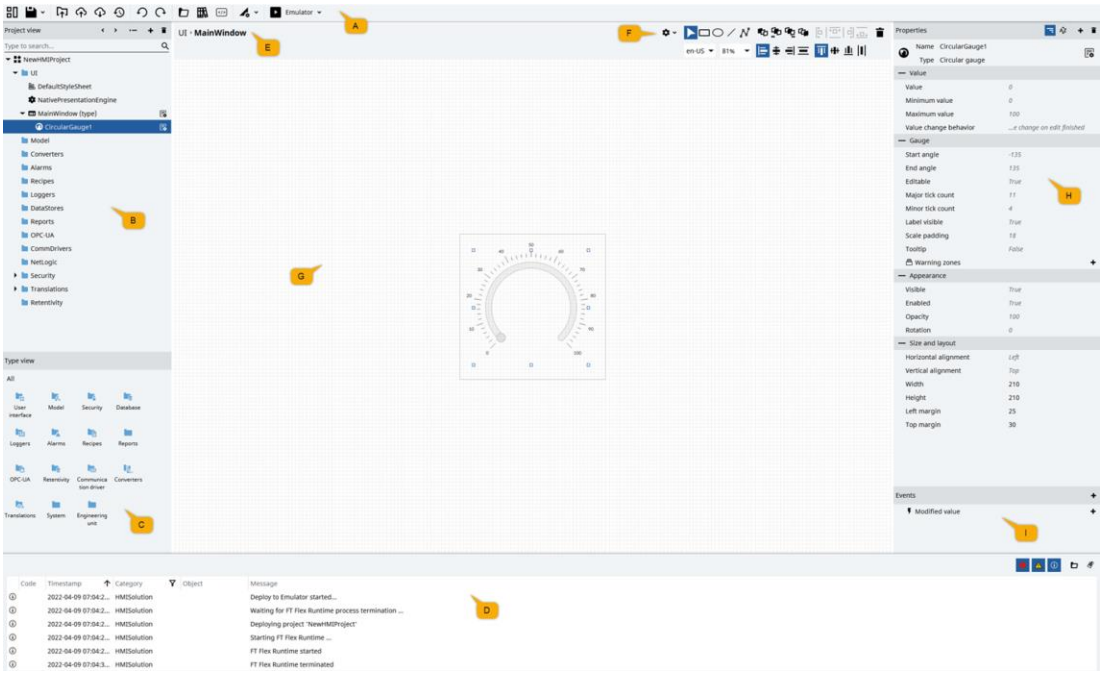
The image has the following software running on it.

- FactoryTalk Optix™
- FactoryTalk Logix Echo V33

No external hardware is required.

FactoryTalk Optix™ Studio Interface

The design environment



Part	Name	Description
A	Main toolbar	See **Main toolbar buttons**.
B	**Project** panel	Project information model to display and set the structure of nodes. The nodes can be organized according to parent/child logics.
C	**Types** panel	Object Types and Variable Types: • FactoryTalk Optix™ Studio native types, grouped in folders according to their purpose • custom project types, located in folders that reflect the project structure visible in **Project**

Part	Name	Description
D	Log Panel	Messages related to the operation of FactoryTalk Optix™ Studio (FactoryTalk Optix™ Studio Output tab), the Emulator (Emulator Output tab), the FactoryTalk Optix™ Applications running and connected to FactoryTalk Optix™ Studio (other tabs with the name of the target).
E	•	Path of the object being edited in the editor
F	•	Specific toolbar for the object type being edited in the editor
G	Object Editor	Graphic editor to set interface objects or to configure other specific object types (e.g. tag importers, recipes, etc.)
H	**Properties** panel	Editor to display and/or set the properties of the selected node in **Project** or in the object editor
I	**Events** panel	Editor to subscribe methods to events that can be generated by the selected object in **Project**, or in the object editor or generated by another selected object.

Main Buttons	Function
	Create, open or close a project.
	Save the project or export the FactoryTalk Optix™ Application.
	Undo the last operation.
	Redo the last operation.
	Open the project resource window for displaying, importing or deleting project files such as images, videos, documents, certificates, etc.
	Open the template library window and template variables. It contains both default and custom libraries.
	Open the .NET project integrated in the FactoryTalk Optix™ project, to integrate functionality through C# code. The .NET project is opened via the default editor configured in the settings.
	Compile, transfer and run the application on the target. In the relative drop-down menu, Configure target opens the window to set the targets.
	End the FactoryTalk OptixTM Application on the target.
	GitHub Integration
	Opens the dashboard page that contains the wizards.

<table>
<tr>
<td>  </td>
<td>
Open the FactoryTalk OptixTM Studio Options window, which contains the following options:
<table>
<tr><th>Label</th><th>Description</th></tr>
<tr><td>Language</td><td>Set the FactoryTalk Optix™ Studio interface locale.</td></tr>
<tr><td>Advanced mode</td><td>Activate the display of advanced log messages in the log panel.</td></tr>
<tr><td>Show feature previews</td><td>Activate the experimental features of FactoryTalk Optix™.</td></tr>
<tr><td>Preferred code editor</td><td>Set the default code editor for the display/change of the NetLogic C# code.</td></tr>
<tr><td>Preferred physical length unit</td><td>Set the default unit of measurement for physical lengths.</td></tr>
<tr><td>Update FactoryTalk Optix™</td><td>Open in the browser the page from which to update FactoryTalk Optix™.</td></tr>
</table>
The following items are available in the relevant drop-down menu:

FactoryTalk Optix™ Studio Options.
Create certificate: opens the window to create OPC UA certificates.

Log configuration level per module: opens the window to configure the maximum information level displayed in the log panel.
</td>
</tr>
<tr>
<td></td>
<td>Open the FactoryTalk Optix™ local manual in the browser.</td>
</tr>
</table>

Buttons in Properties	Function
	Go back in the panel selection history.
	Go forward in the panel selection history.
	Divide the **Project** panel into two panels.
	Create a new node inside the selected node.
	Remove the selected node and the nodes it contains.
	Group the child nodes: hide them from the **Project** and make them non-editable.

Buttons in Properties	Function
	Release the child nodes of the group: display them in the **Project** and make them editable.
	Only for some object types. Open the object in the dedicated editor.

Log Panel Buttons	Function
	Show/hide the error messages of maximum severity.
	Shows/hides the error messages of minimum severity.
	Show/hide information notices.
	(in advanced mode only) Show/hide additional information about the error messages or information notices displayed.
	(in advanced mode only) Show/hide additional debug information about the error messages or information notices displayed.
	Open the logs folder of the running element (FactoryTalk Optix™ Studio at design time, FactoryTalk Optix™ Application at runtime or **Emulator**).
	Delete messages from the panel.
	Open the selection window of the modules to display in the **Category** column

Information in the log panel

Column	Column name	Description
1	•	Information level of the messages: •  = error of maximum severity •  = error of minimum severity • = information notices

Column	Column name	Description
		• (in advanced mode only) V1 = additional information about the error or alert • (in advanced mode only) V2 -= additional debug information about the error or alert
2	Code	(Only for FactoryTalk Optix™ Studio native messages) Message ID code
3	Time	Message timestamp
4	Category	Category of FactoryTalk Optix™ Platform message or module that generated the message.
5	Object	•
6	Message	Message content

Editor Buttons

The buttons displayed change depending on the element being edited.

Properties Panel Buttons

Button	Function
	Organize the properties in groups.
	Sort the properties in alphabetical order.
	Add a property to the node.
	Remove a property or reset a property to its default value.
	Only for some object types. Open the object in the dedicated editor.

Events Panel Buttons

Button	Function
+	In the **Events** panel header: add an event of another object. Near an event: associate a method with the event.
🗑	Delete the selected method.

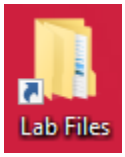
Common elements

Element	Function
✎	Change the name or value of the element.
🔗+	Open the dynamic link window to set the value of the corresponding field with a dynamic link.
🔗	Open the dynamic link window to edit the existing link.
⌄	Open the specific dropdown menu for the relative button.

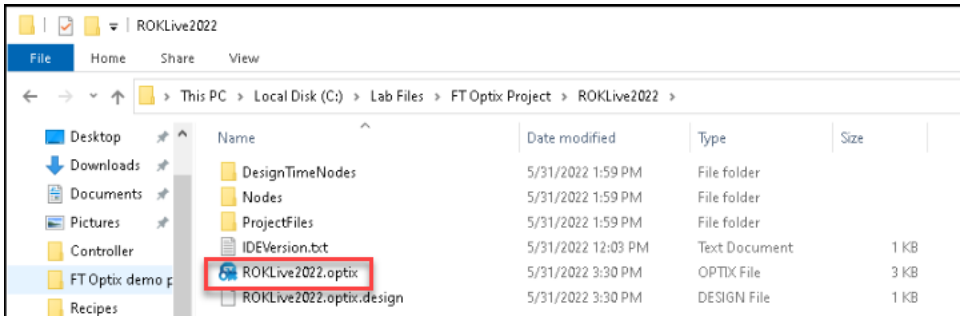
Tags and connecting to controllers

In this section of the lab you will see the how many different controller drivers are natively available in FactoryTalk Optix™ and add and control a tag from a Rockwell controller.

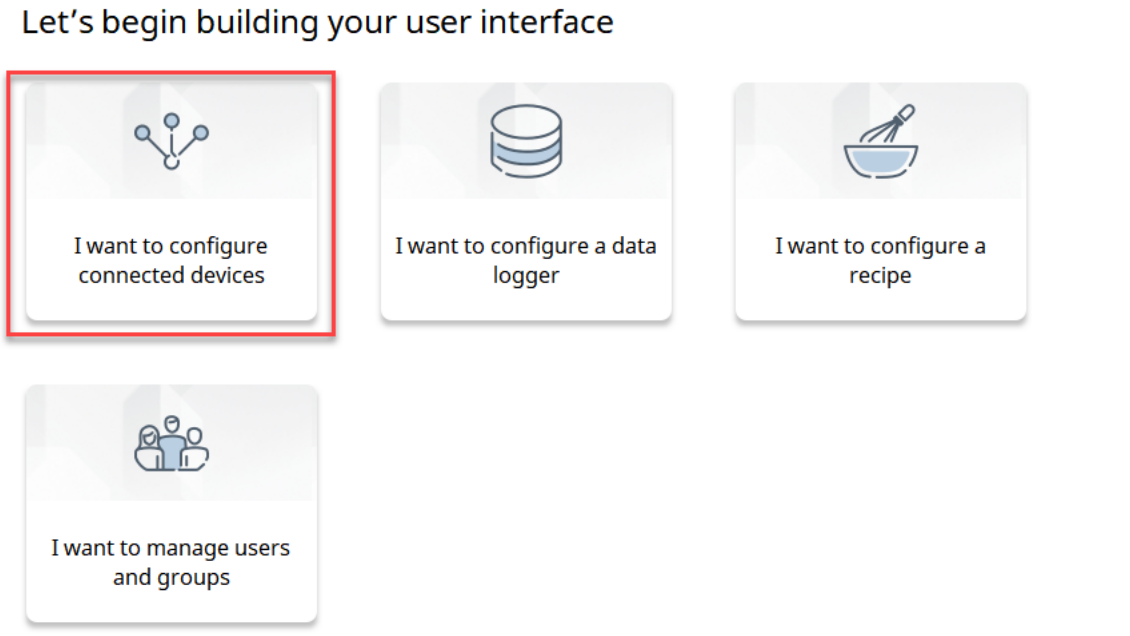
1. *Double click* on the **Lab Files** shortcut on the Desktop.



2. *Double click* on the **FT Optix Project > ROKLive2022** Folder and then *double click* on the FactoryTalk Optix™ project **ROKLive2022,** this will open the project in FactoryTalk Optix™ Studio.

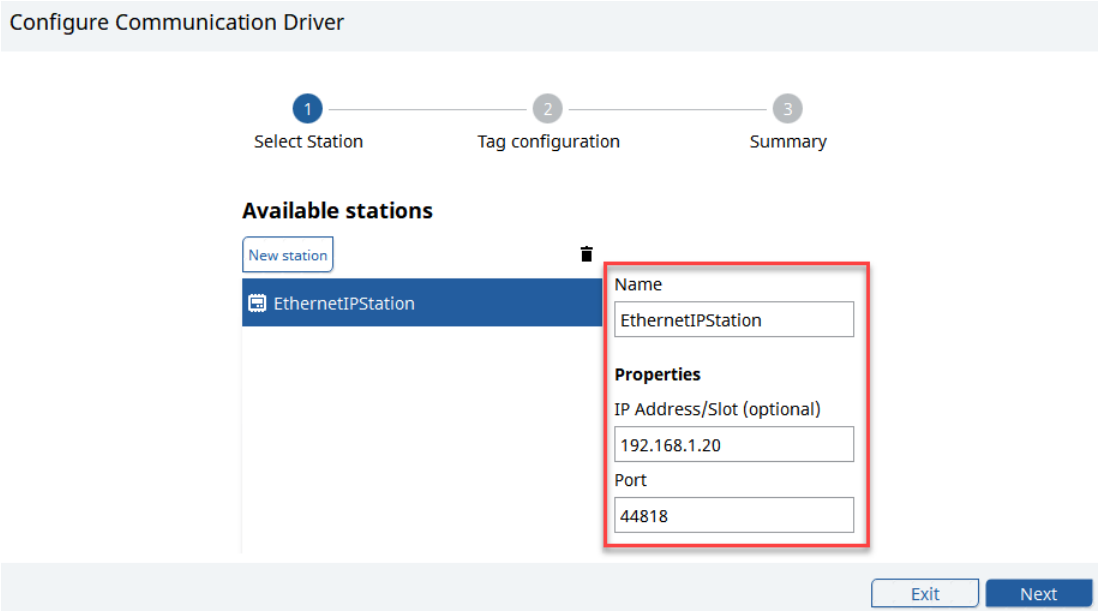


3. Once the project is open, you will see wizards, which will be used throughout the lab. For this section, you will use the I want to configure connected devices.



The wizards can always be found in the menu by clicking on 

The first step in the Configuration of the Communication driver is to configure a Station. In this lab the Station has already been created.



Name : This is a name that you can give your controller.

IP Address/Slot: This is the unique IP address of the controller. Below are some examples of the IP addresses that you could use.

Port : This is the default port used in the controller.

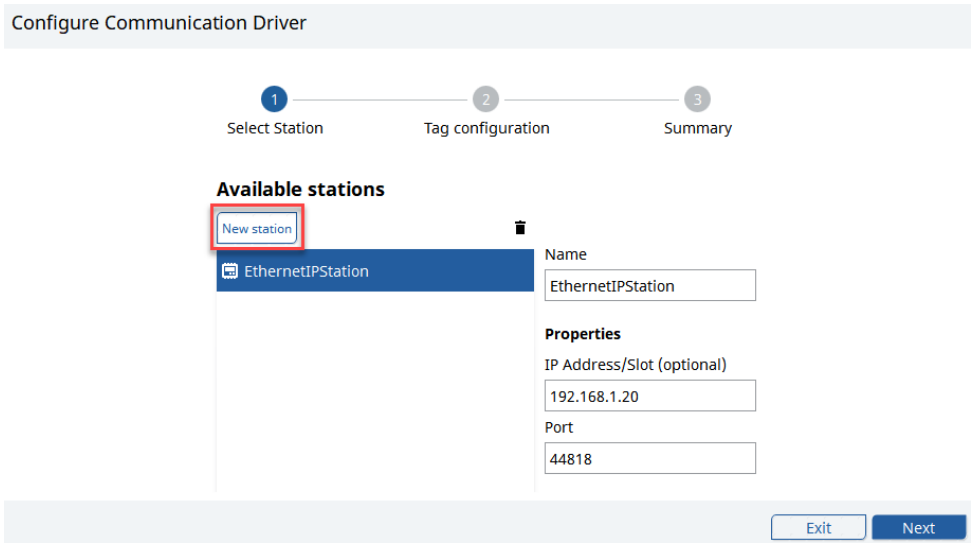
If you want to connect to controller in slot 0 through the Ethernet module in slot 3, you need to use the following address : 192.168.1.50/1:0

If you want to connect to controller in slot 1 through the Ethernet module in slot 3, you need to use the following address : 192.168.1.50/1:1

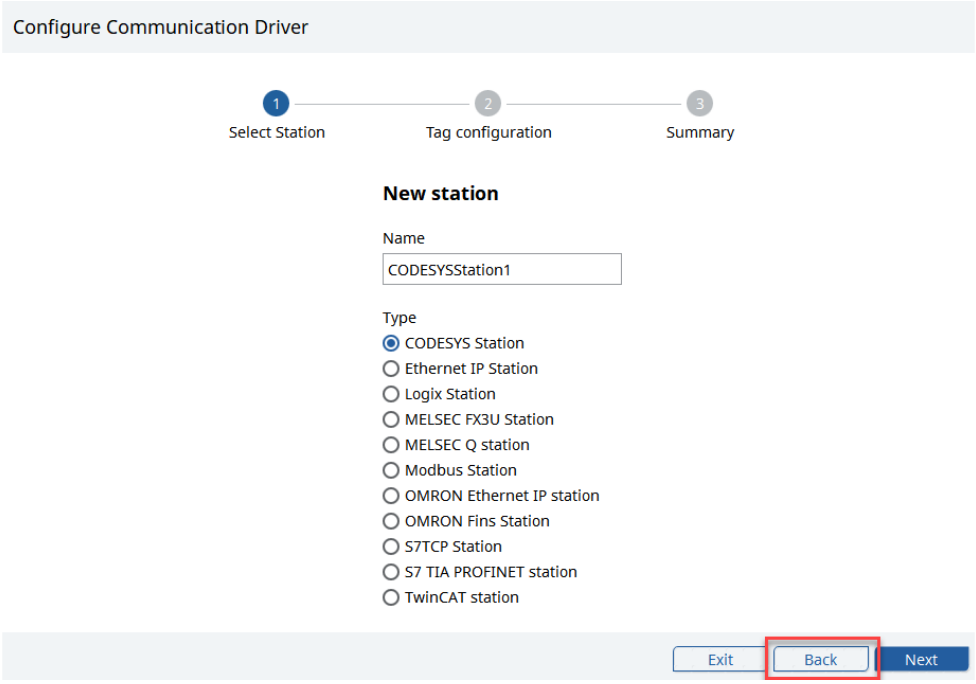
If you want to connect to controller in slot 2 which has an IP address, then you can use IP address : 192.168.1.100

If you don't have an ethernet module in your chassis but you want to connect to the controller in slot 0, you can go through the front port of the L81E controller by using the following IP address: 192.168.1.100/1:0

4. *Click* on New Station



For this lab you will not add another station but you can see the list of native drivers that can be added to FactoryTalk Optix as well as communicating via OPC UA. Click on Back to return to the previous screen.



Read about OPC UA Connectivity

Besides these stations/drivers that can be added to FactoryTalk Optix, the software can also act as OPC UA Server and as an OPC UA client that can connect to 3rd party OPC UA servers.

OPC UA

Introduction

Thanks to compliance with the OPC UA (OPC Unified Architecture) standard, with FactoryTalk Optix™ *it is possible to create applications that communicate with any OPC UA client/server. Also, devices equipped with suitably configured* FactoryTalk Optix™ *Application can operate as either client or server.*

FactoryTalk Optix™ Studio allows nodes to be imported from an OPC UA server at runtime or at design time to create the desired application logic, similar to what is possible with the PLC tag importer.

To create these functions, the two OPC UA Server and OPC UA Client objects are available.

Server OPC UA

The OPC UA Server object publishes the project information model nodes at runtime. By default, it publishes all project nodes, but it is possible to select which nodes to publish and to which users. This object is necessary to allow an OPC UA client to read/write from/to the server, to invoke methods or to listen to events.

It is possible to configure one OPC UA Server object at most in a project.

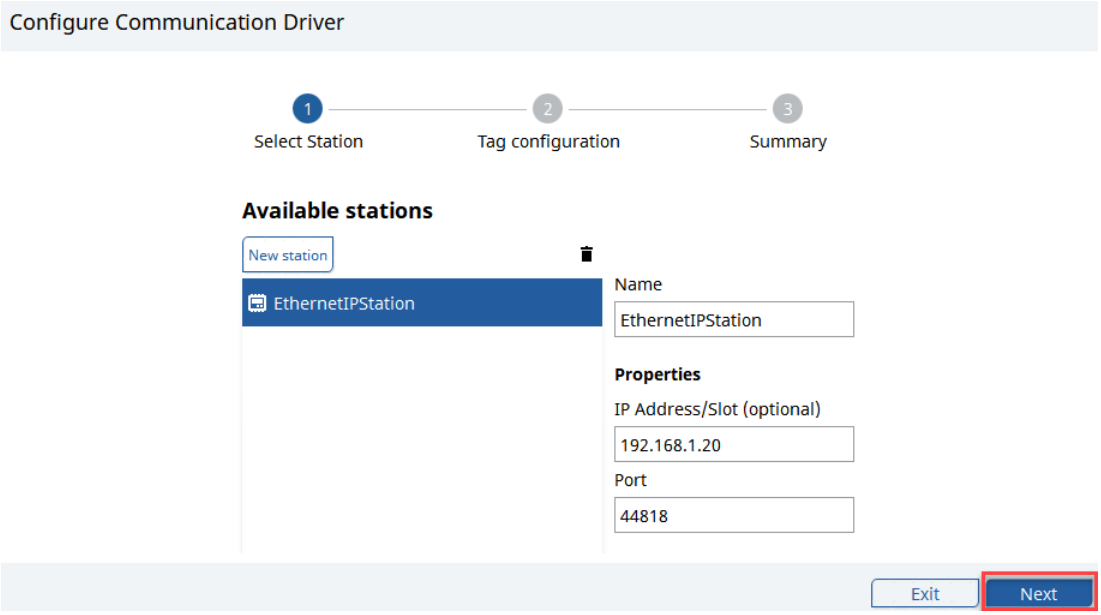
Client OPC UA

The OPC UA Client object allows communication with an OPC UA server. Specifically, it can access nodes published by the server to read/write, invoke methods, or listen to events.

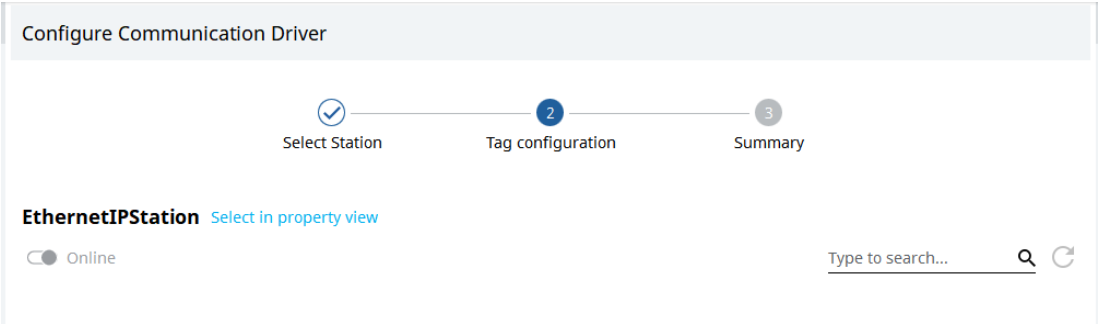
It is possible to specify the nodes of interest to the server, which can be imported at design time or at runtime. An OPC UA Client can, for example, read some specific object variables, or listen to events generated by alarms and possibly acknowledge all the active alarms, or import the users configured on the server.

It is possible to configure an unlimited number of OPC UA Client objects in a project.*

1. *Click* on ***Next***.

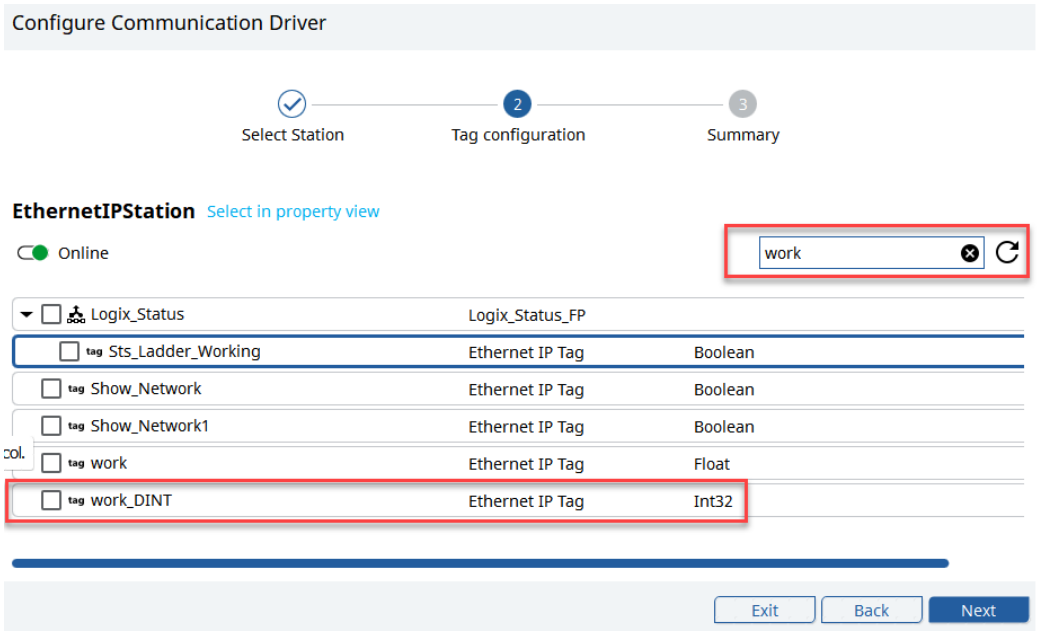


2. Wait while the driver connects to the controller and uploads the tags.

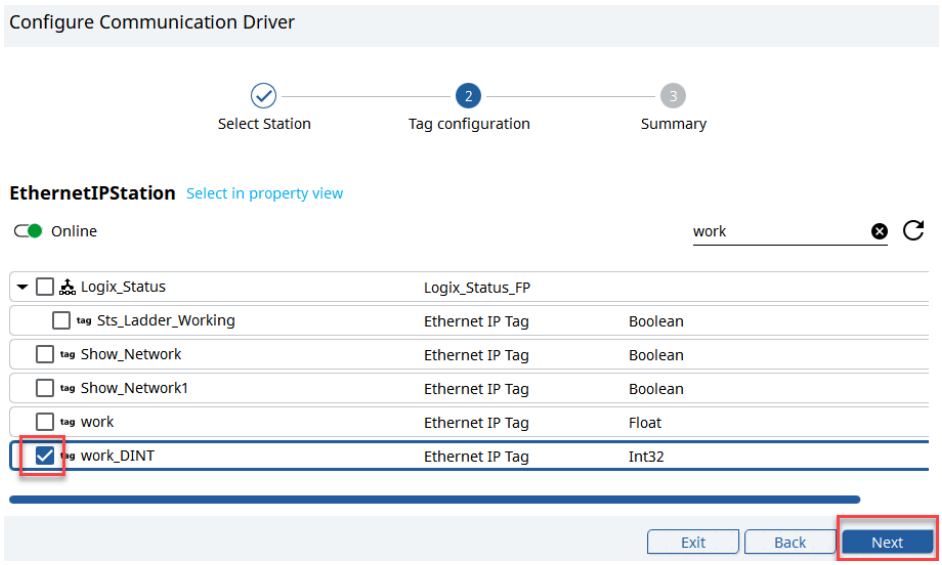


In case you don't have a controller, you can develop in Offline mode through an L5K, L5X or ACD file.

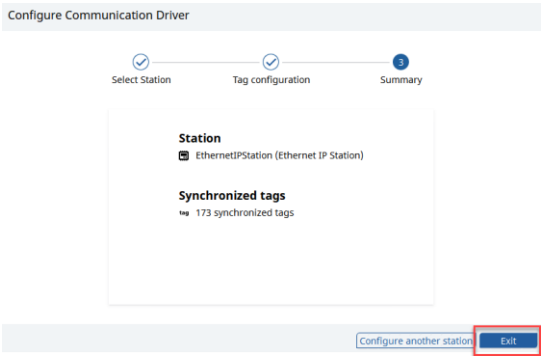
3. When the list of tags appears, you can look in the list for tag **work_DINT** or you can type the name or part of the name in the **search** bar.



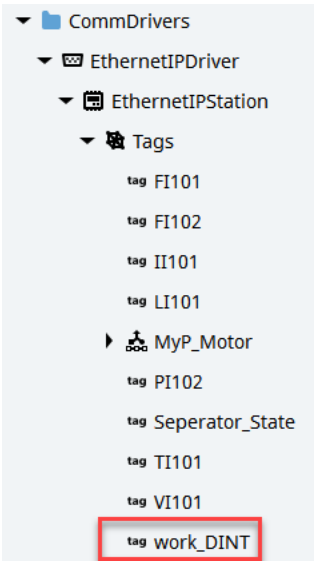
4. *Check* the Checkbox in front of the tag and select ***Next***.



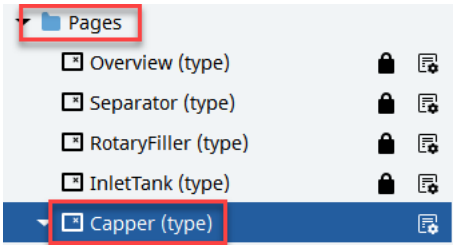
5. The tag is now synchronized, select ***Exit***.



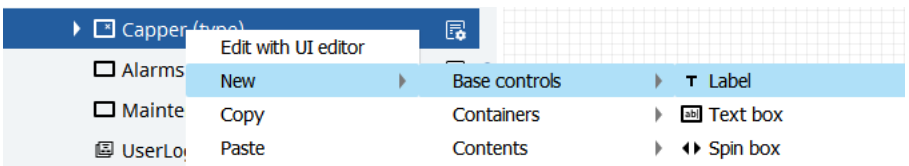
6. Great work, you have added a controller tag into your project. Now you will add the tag to the main screen and manipulate it.



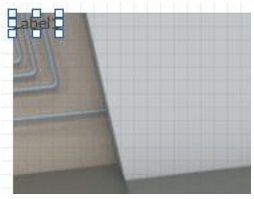
7. Go to **UI > Pages** and *double click* on ***Capper (type)*.**



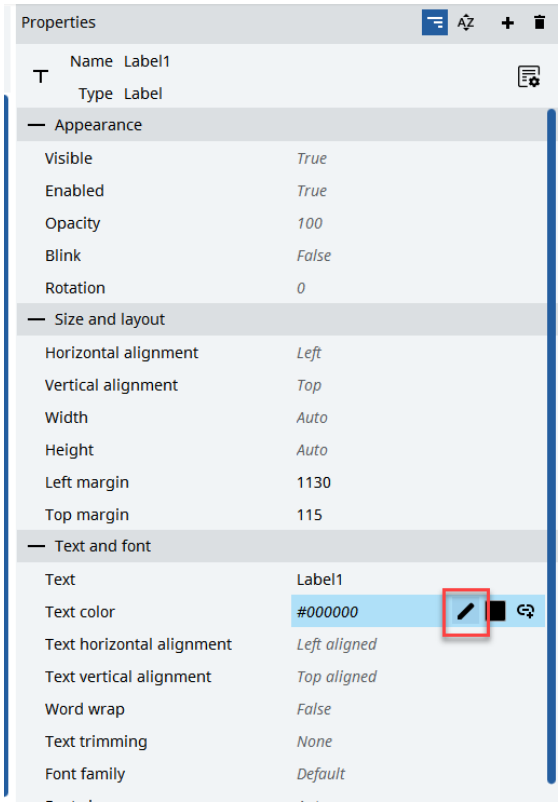
8. ***Right Click*** on the **Capper** display and select ***New > Base controls > Label***



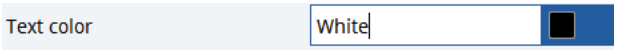
9. The **label** will appear on the top left corner of the **Capper** screen, make sure the label is selected.



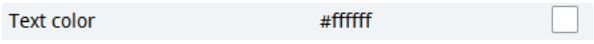
10. On the right side you see the properties of the label, in here *select* the Text Color and then click on the  icon. If you don't see this property then scroll down in the property section.



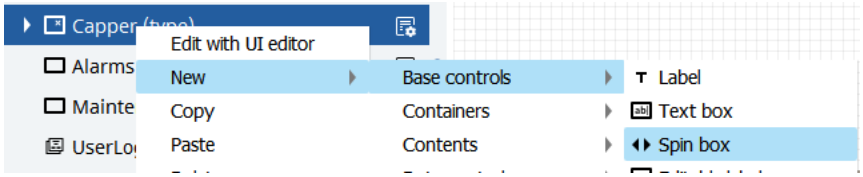
11. In the box that is highlighted, type in ***White*** and *press* **Enter**



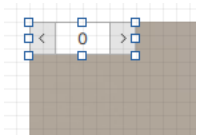
12. The code *#ffffff* appears in the box and next to it the box is colored *white*.



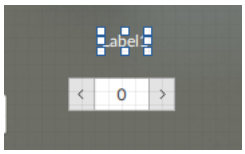
13. *Right Click* on the **Capper** display and select ***New > Base controls > Spin Box***



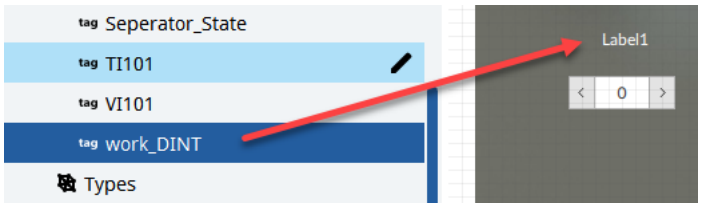
14. The **Spin Box** will appear on the top left corner of the **Capper** display.



15. Move the objects and arrange as shown below.



16. Now expand the Folders **CommDrivers > EthernetIPDriver > EthernetIPStation > Tags**, *Click* on the tag **work_DINT** (keep clicking) and *drag* it to the **label** object.

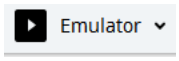


17. The **label1** text will automatically change in **#####** .

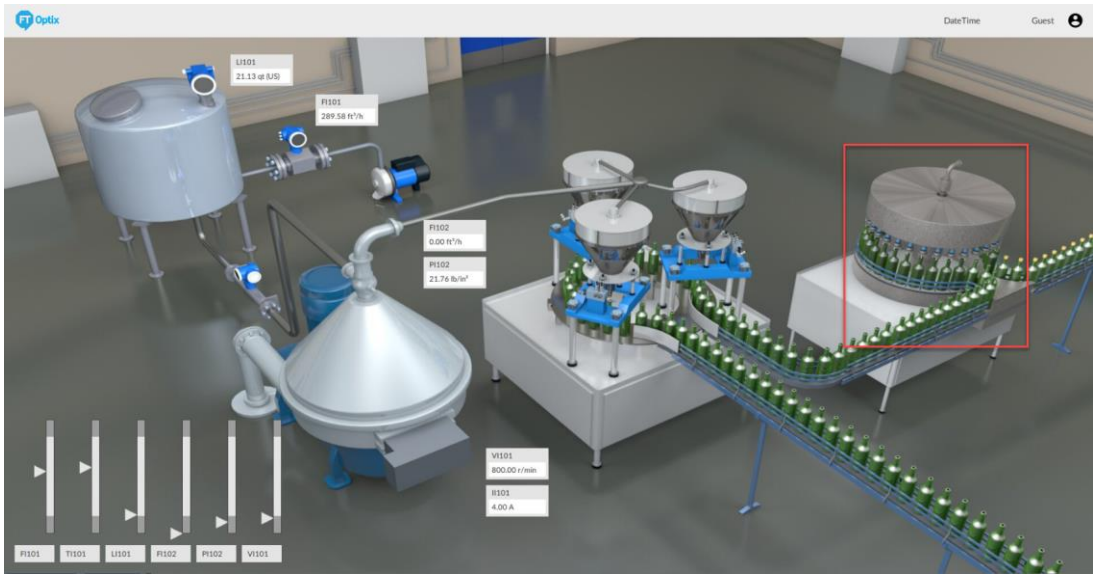
18. Repeat **step 16** to associate the tag with the **Spin box**. The **Spin Box** will also show now **#####.**

19. Now that you have linked the tag **work_DINT** to the **label** and the **Spin Box**, you will run the application to see if it works.

20. In the top bar, Click on ***Emulator***.



21. On the **Overview** screen, *click* on the ***Capper***.



Important note

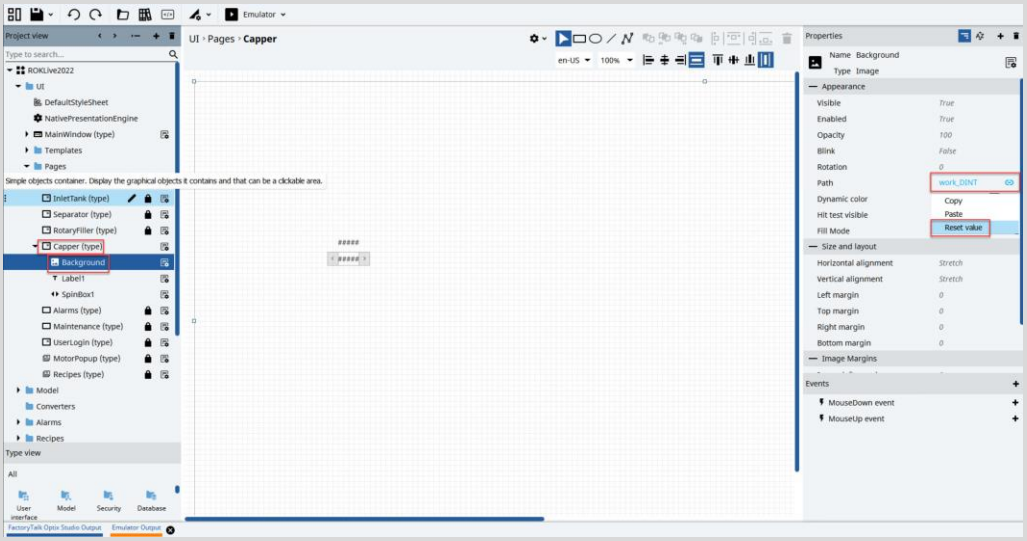
If you click on the Capper screen and it looks like the screen below, without a background image.

By dragging the tag to the objects on the screen, you dragged and dropped the tag accidently onto the background and not the object.

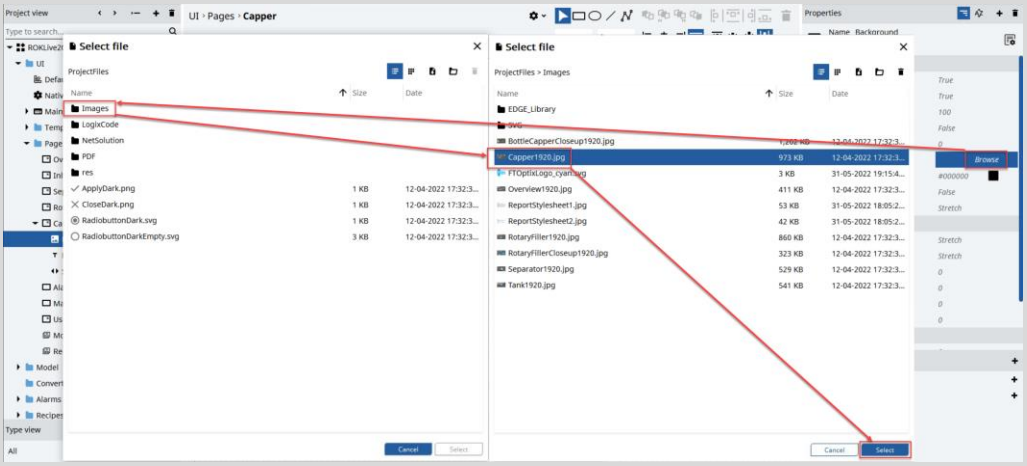
In effect you have set the browse path to the tag name.

These are the steps to get your background image back.

Go to the **Background** object on the **Capper** screen, in the properties, locate the **Path**, *Right Click* on the **work_DINT** and select ***Reset Value***.

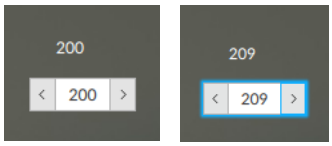


Now *Browse* to the **Capper1920.JPG** file in the **Images** folder.

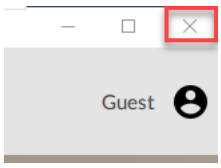


22. The **label** and the **Spin Box** will show a value. Using the **Spin Box** you can change the value in the Label.

    When you click on the value in the spin box, you can input a value.



23. *Click* on the **X** in the top right corner to close the **emulator**.



Display construction

In this section of the Lab you will learn the fundamentals of constructing Displays with FactoryTalk Optix™ . You will learn about containers Widgets, Objects, Types, the Information model, Aliasing and Instances and how they are used in FactoryTalk Optix™,.

Read about Graphical objects organization

Introduction

The objects that make up the user interface of a FactoryTalk Optix™ *Application are organized in the **UI** folder.*

*All the graphical objects directly children of the **UI** folder, or of one of its subfolders, are object types; they are then available in the **Types** panel to create instances.*

Hierarchical organization

Typically a user interface is organized in a hierarchical way, that is the different elements that compose it are organized in a tree in a more specific way according to their function.

The hierarchy of graphical objects within the information model is determined by graphical object containers, the only ones that can contain other graphical objects. The position of the objects in the information model determines their position in the layout, at both design time and runtime.

Window

The Window object is the root container graphical object. It must be present when you want to generate a FactoryTalk Optix™ *Application with a user interface. Inside, all the objects that make up the user interface and the related logics are organized.*

It is possible to create several Window objects, to associate with different Presentation engines. For this reason, a Window object can only exist as a type.

Panels

A panel is a type of container that aggregates several relevant graphical objects. The navigation in a FactoryTalk Optix™ *Application typically takes place between the different panels set up.*

User interface navigation

Navigation within a user interface can be created both with graphical objects dedicated to navigation (for example the Navigation Panel and Dynamic Panel objects) and with customized logics within NetLogic.

The objects dedicated to navigation include interaction mechanisms and runtime logics to navigate between different set panels. Such objects at runtime create and delete GUI nodes depending on the user's navigation.

For the description of the objects for navigation refer to the Reference on objects and variables section of this manual.

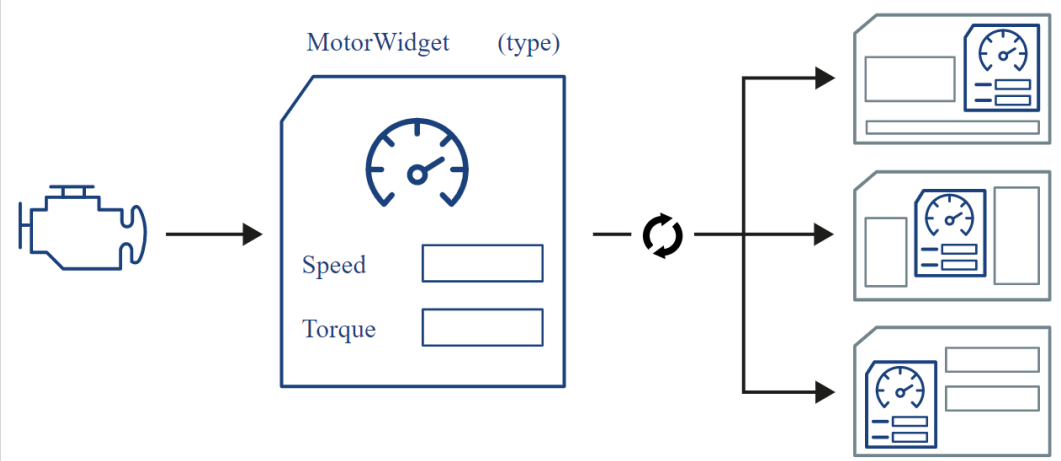
Reusable graphical objects: widget

Typically, in a user interface it is necessary to consistently present - at runtime - and efficiently manage - at design time - structures of nodes that are identical but associated with different data. To do these types of graphical objects are needed with which to create identical objects. These types are conventionally defined as widgets. A widget can be a single graphical object or a group of graphical objects, designed for a specific purpose of display in the user interface and which includes all the necessary logic, created with dynamic links and/or NetLogic.

Note

the association with different data in the various instances of the same widget is typically done through an alias, defined in the widget.

For example, a MotorWidget widget might consist of a panel containing a circular Gauge object to display speed and two labels to describe the numeric values of speed and torque. The widget can be created at design time or at runtime for each motor whose values you want to display. Since the widget is an object, any changes to the widget structure and formatting propagate across all instances.



Concepts of layout

Introduction

The layout of a user interface in a FactoryTalk Optix™ *Application depends on the following elements:*

the organization of the graphical objects in the information model

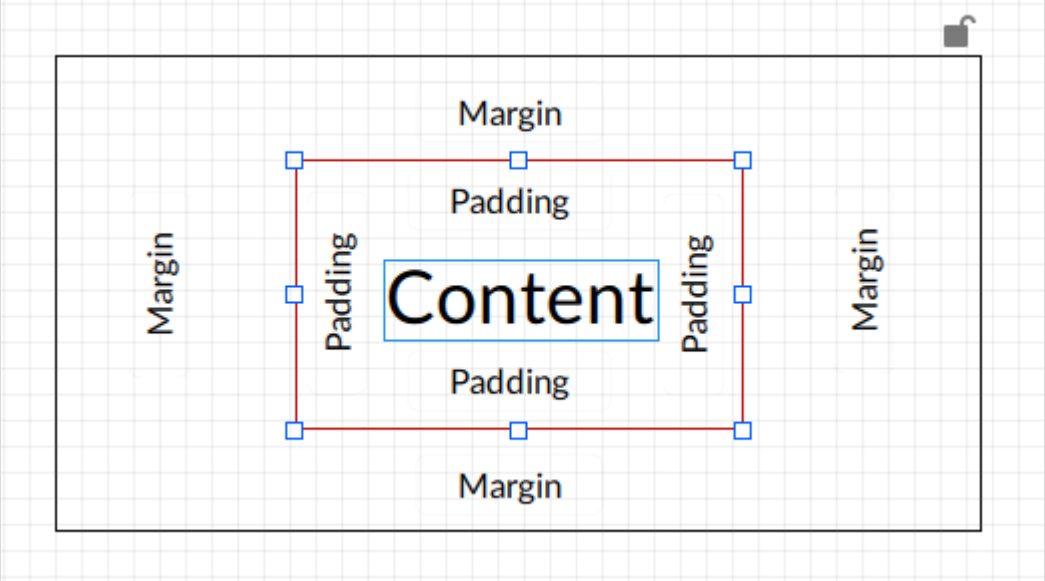
the value of the positioning (alignment and margins) and sizing (width and height) properties of the graphical objects

the order of the graphical objects on the z axis

the size and aspect ratio of the screen on which the interface is displayed.

Margins, padding and borders

Below is an example of ***Text box*** *object to present the meaning of the terms content, margin, padding and border:*



***Content**: Content text*

***Padding**: space between the content and the edge of the object. Both horizontal and vertical padding are values relative to the height of the text contained (for example, if the padding is 60%, a label with text 20 pixels high has padding of 12 pixels).*

Note

*the padding is present only in the Text box, Button, Spin box, Data grid, Drop-down and List **selection** objects.*

***Edge**: edge of the object, in the example in red*

***Margin**: space between the edge of the object and the edge of the parent object (container)*

Note

The padding, edge thickness and edge corner radius, if any, are global style properties defined in a style sheet.

Positioning of the objects

The positioning of a graphical object is determined by its alignment on the x and y axes and by its margins with respect to the sides of the container object. The margins to be set depend on the alignment set, or the value of the Horizontal alignment and Vertical alignment properties.

The position of objects on the x and y axes in the layout is always relative to the position of the container. If a container is moved, then all the child objects are also moved.

The position of objects on the z axis compared to other child objects in the same container is determined by its position in the information model. Objects further down in the information model are higher up on the z axis.

For a description of all properties related to the positioning, refer to the properties of graphical objects in the Reference on objects and variables section of this manual.

Size of the objects

The size of a graphical object is defined via the Width and Height properties, only if the alignment on the x and/or y axes is left, right or center.

*If the value(s) of the Width and/or Height properties is/are deleted, the two dimensions take on the Auto value. In this way, for some objects the size is determined by the intrinsic content (for example the text in a Text Box object) or by other contained nodes (for example graphical objects in a **Panel** object). For example, the height of a Text box object with Height = Auto equals the height of the text, plus any padding set in the style sheet.*

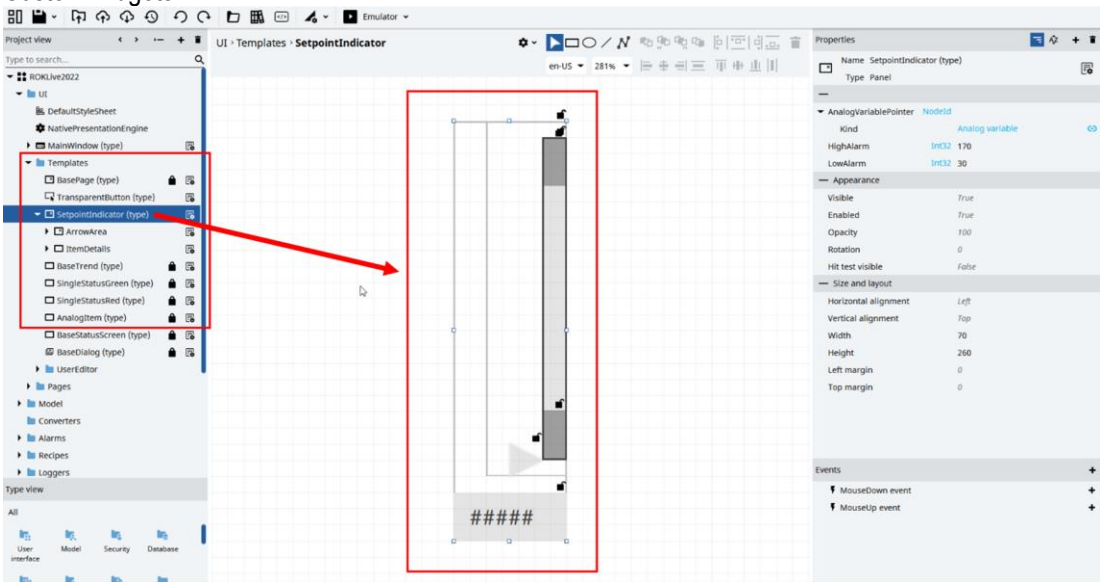
If the alignment on the x and/or y axes is set to Adjust, the width and/or height assume the dimensions of the container, minus any margins. This setting makes the size of the object dynamic relative to the size of the container.

Note

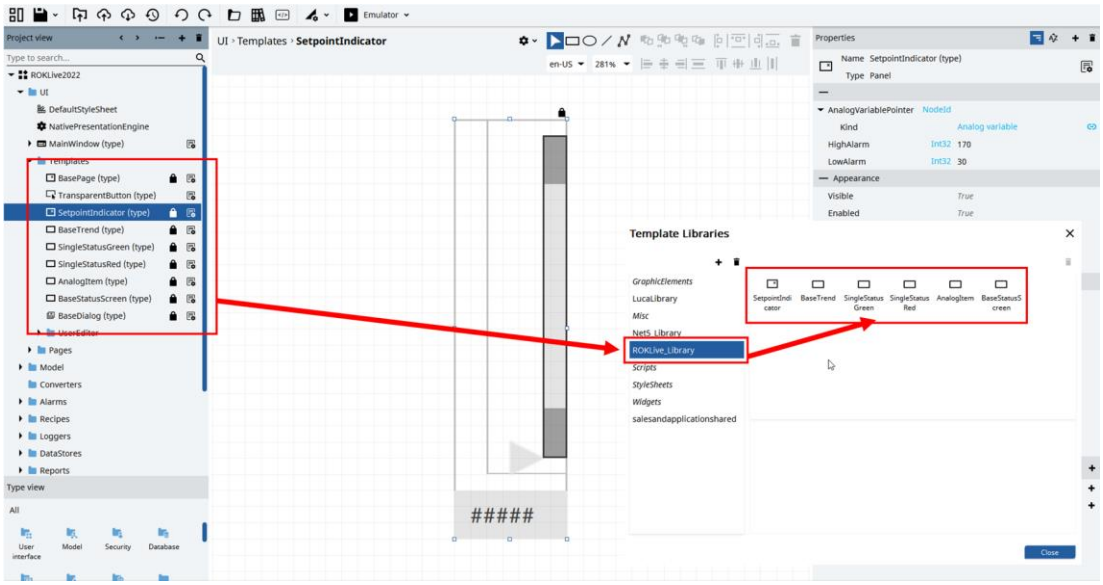
It is not possible to scale in percentage the size of an object inside a container with respect to the size of the container itself.

Widgets

Custom widgets:



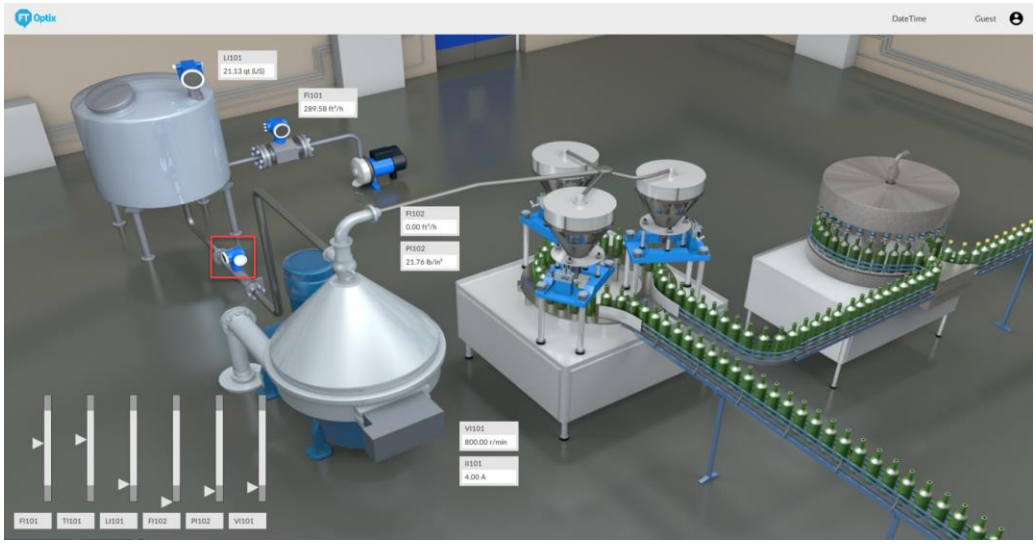
These included widgets show the full versatility of FactoryTalk Optix™, you can consider this IDE as a blank page with a fancy pencil, you can design and develop anything you want using basic elements. In this lab, we provided some indicators inspired from Studio 5000 and replicated them using a mix of panels and rectangles, animation of indicating arrow is done by passing a variable value to the “bottom margin” of the arrow itself. Using aliases you can in fact pass any type of object to a type and do some elaboration depending of the status/value of the alias. When you finished drawing these custom widgets you can drag them into your personal Template Library and store them for future projects or to share it with other colleagues.



These items have been included in your start project to help you while proceeding in the lab, same as you would do by importing items from the Template Library.

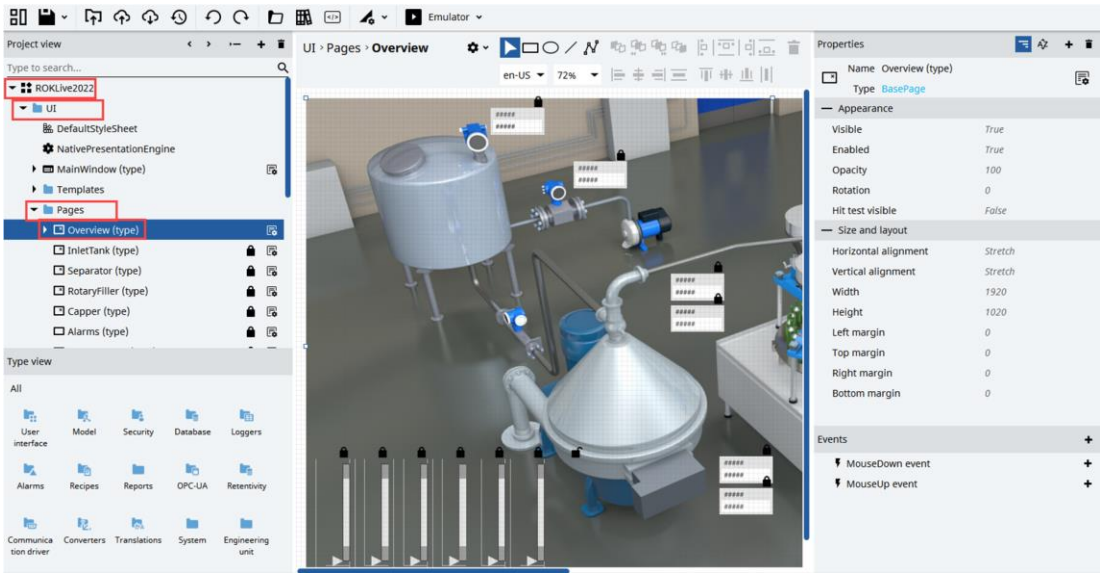
Adding an AnalogItem widget to the **Overview** display.

This widget will show the current value of an AnalogVariable with localized measurement units and its DisplayName, you will now add the current fluid temperature to the project.



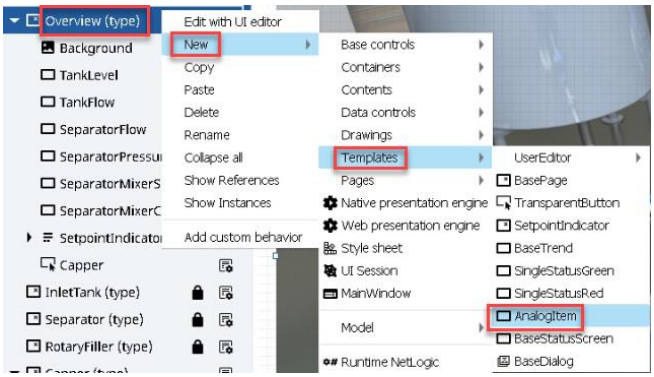
1. Create a new instance of “AnalogItem” widget.

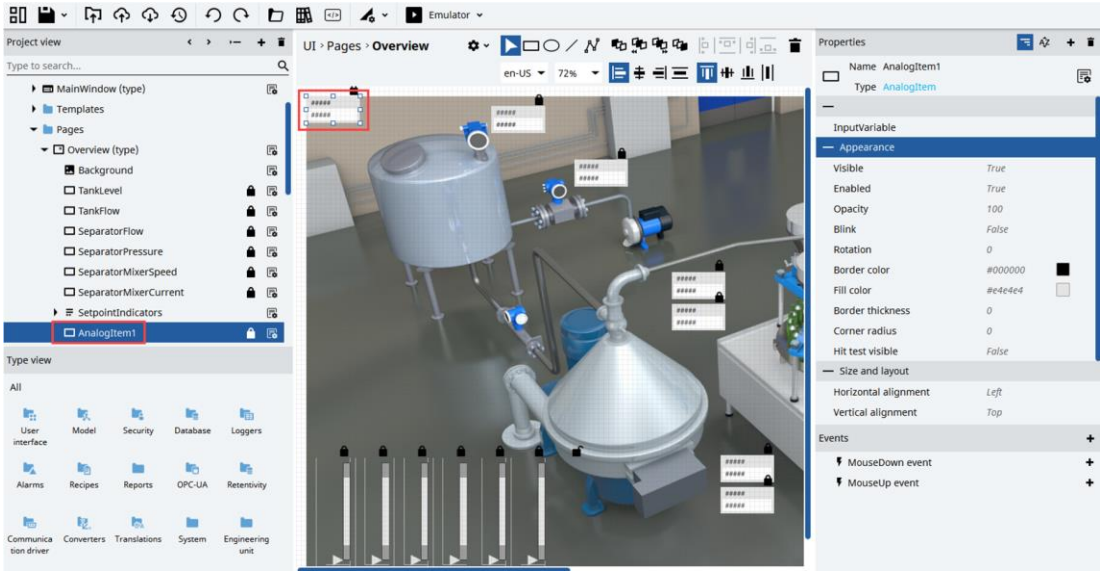
   Browse to **UI > Pages** and *double click* on **Overview** to open the display editor.



2. Now you will add the custom widget to the Overview screen.

   Right click on **Overview** and *hover* to **New > Templates** and *click* on **AnalogItem**, the new object will appear in the project tree and in the editor window.

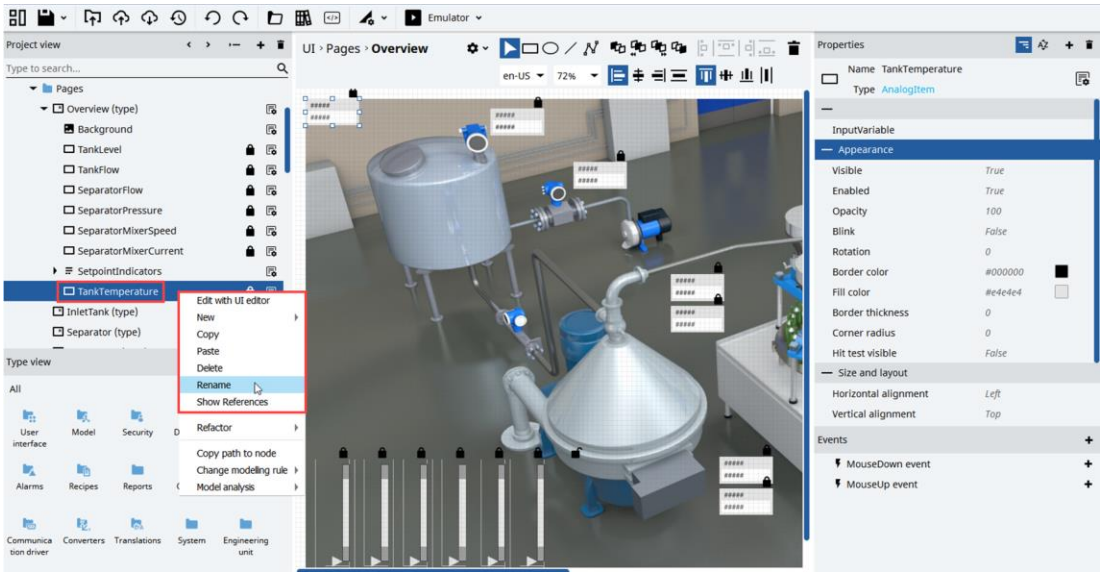




3. Give a meaningful name to the widget.

   Rename this object to **TankTemperature** by *right clicking* it, giving useful names will help when browsing in big projects.

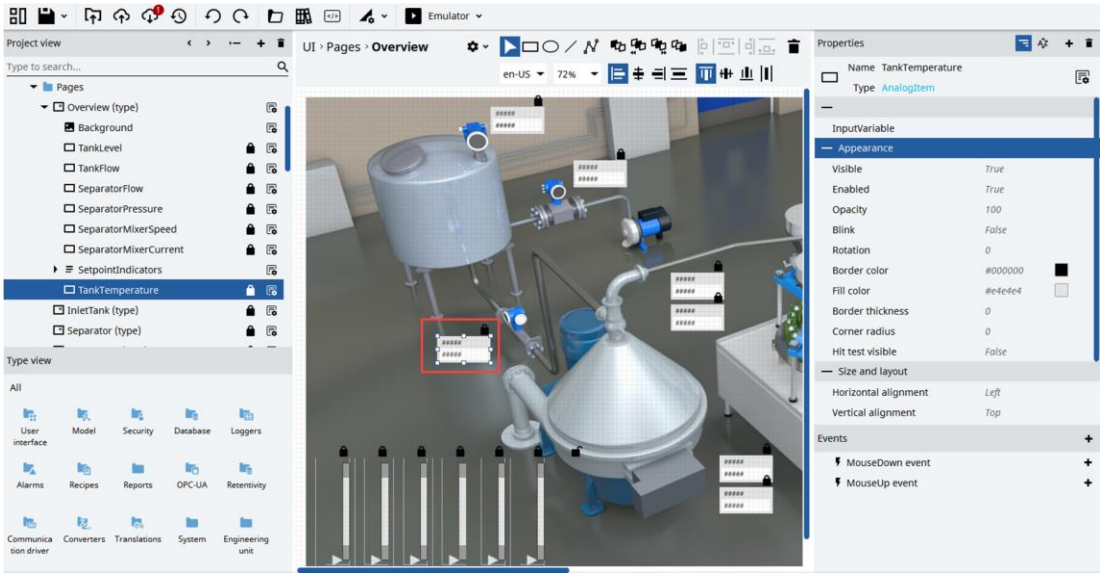
> When naming items pick a name that may hep you find that same object later, avoid using base item names (e.g. no SpinBox, Button, …) as it may be confusing and creating problem to the project
>
> Even if it's allowed, avoid using same names in different containers, it may create confusion when browsing the project, try to use unique names.



4. Place the widget in the desired location.

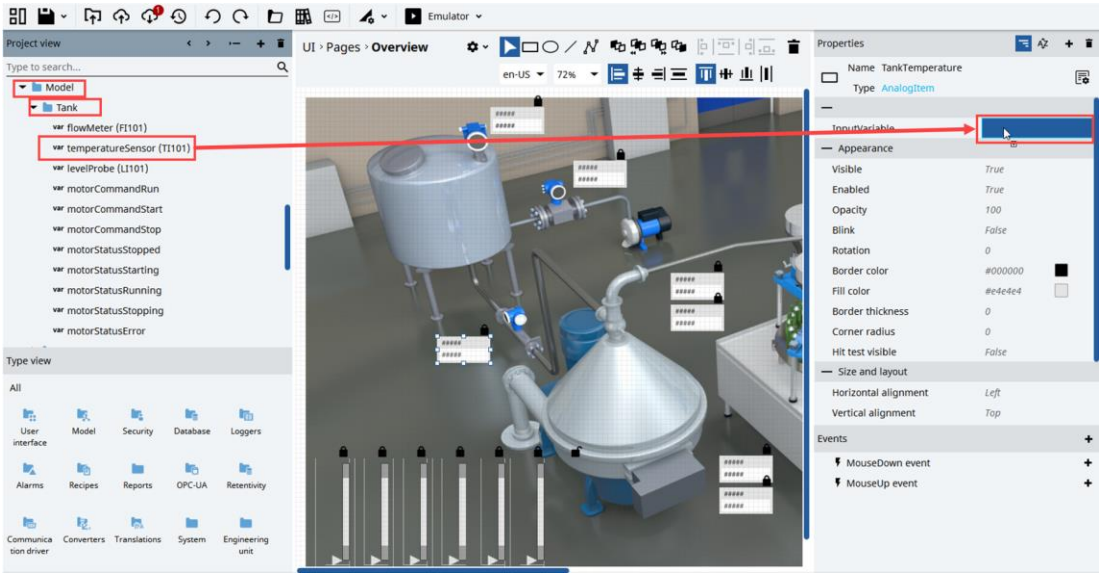
   Now *drag and drop* the widget close to the temperature sensor in the process pipe.

> Items can be dragged in desired location in the editor panel or manually placed setting padding, alignment and margins using the properties panel in the right side.

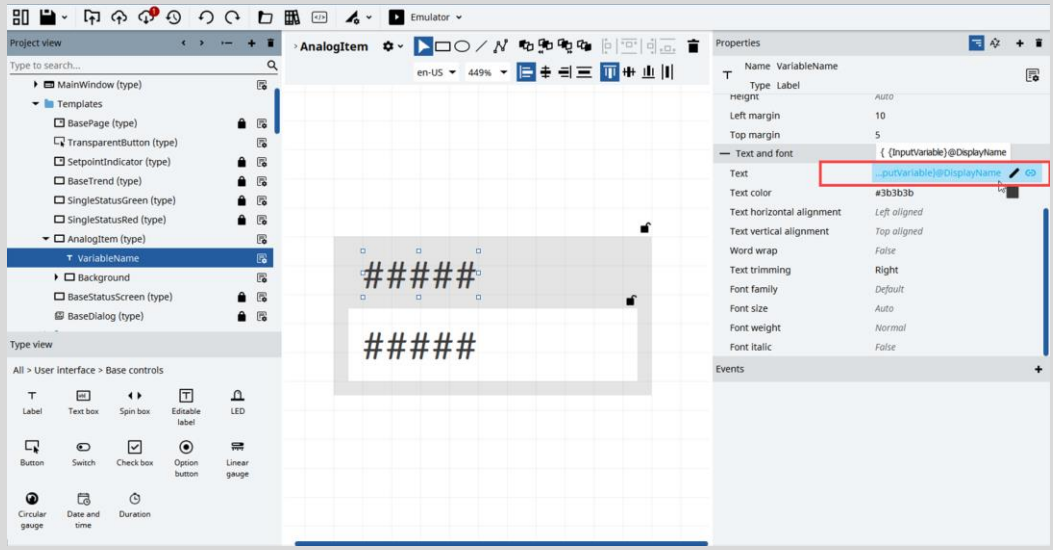


5. Now you will configure the object, *click* on the object and *drag* the desired variable to the **InputVariable** parameter, this will create a “Dynamic Link” between the two objects

   Click on the **TankTemperature** widget to select it, browse in the project tree to **Model > Tank** and *drag* **temperatureSensor (TI101)** to the **InputVariable** field.

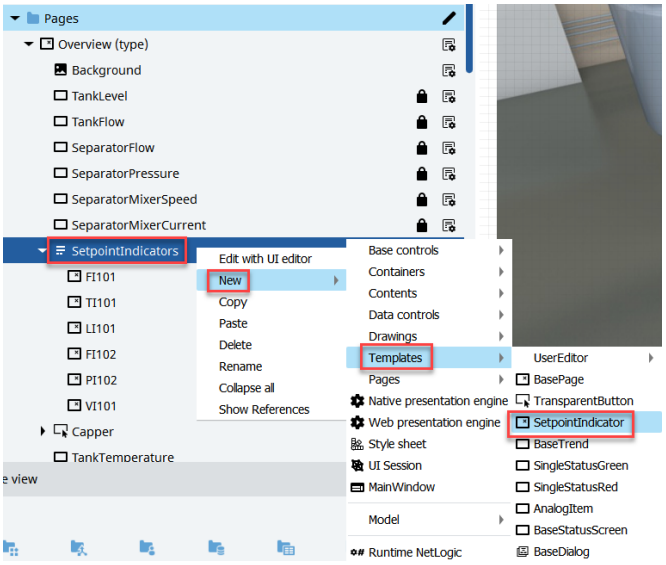


InputVariable field is called “Alias”, an alias is an argument passed to an object that can be used to do some processing inside that object, in this widget you use InputVariable alias to extract DisplayName, value and EngineeringUnit and show them in the two textboxes



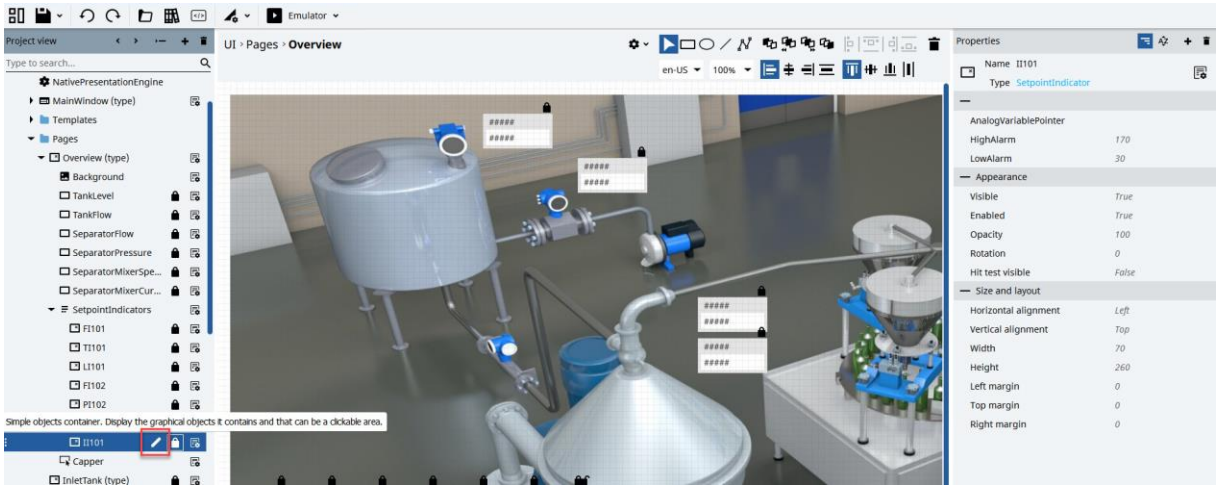
Add a new SetpointIndicator to Overview screen.

1. *Expand* the *Overview* display and browse to **SetpointIndicators** horizontal layout and expand it, you are going to add here a new widget by *right clicking* on the container and then **New > Templates > Setpoint Indicator**.



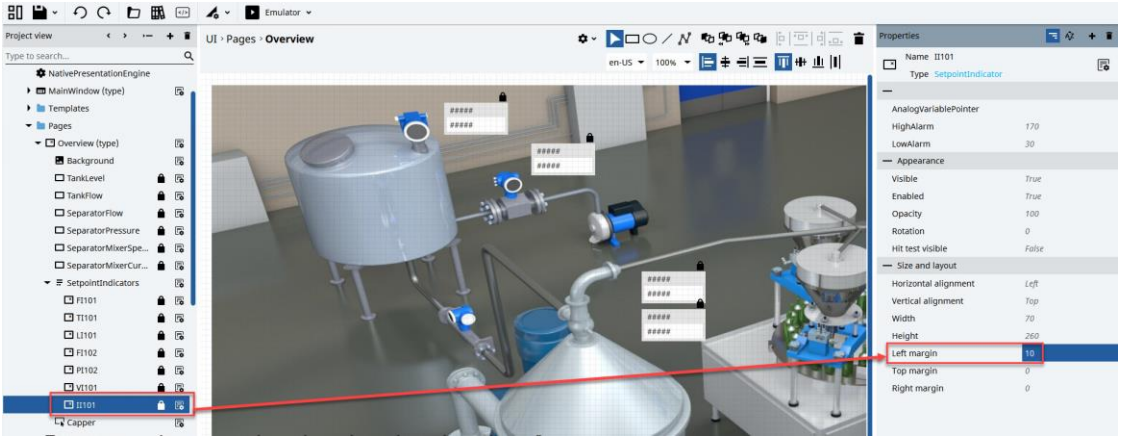
2. Set the name to **II101.**

   Hover to the new item name and *select* the pencil icon to open the editor, change the name to ***II101***.



3. Set the alignment.

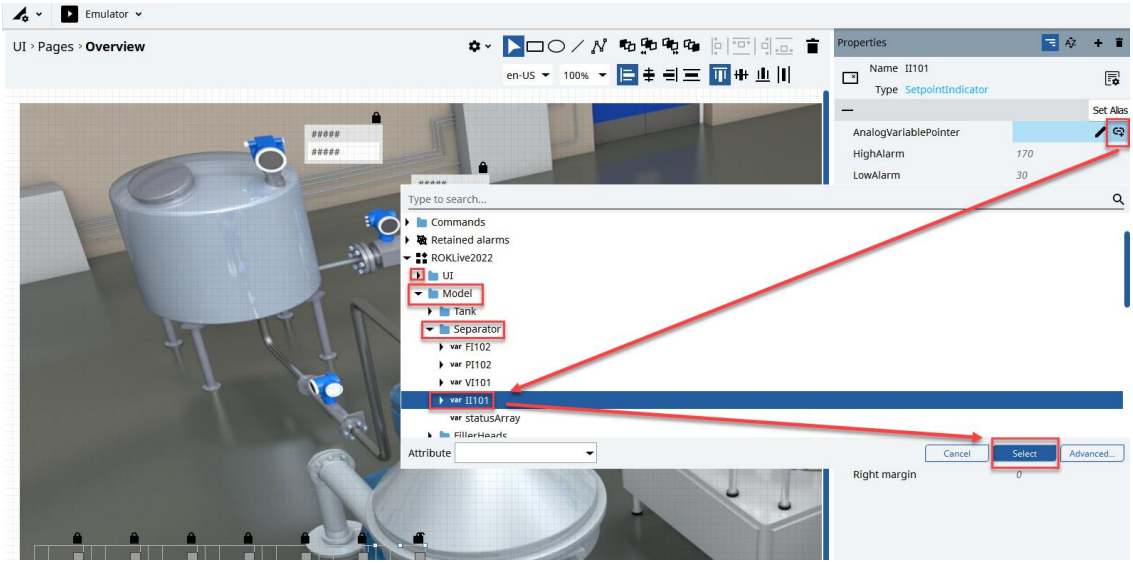
   The item will automatically be placed at the end of the HorizontalLayout, change the **Left Margin** to ***10px*** to space it a little from other objects.



4. Configure the widget.

   You will now set a DynamicLink to the **II101** variable in the **Model folder**, *click* on the DynamicLink editor in the AnalogVariablePointer parameter to open the project browser, Collapse the UI folder, expand the Model folder and navigate to **Model > Separator** and *click* once to **II101**, *click* **Select** to create the link. Source variable name will appear with its BrowseName in brackets.

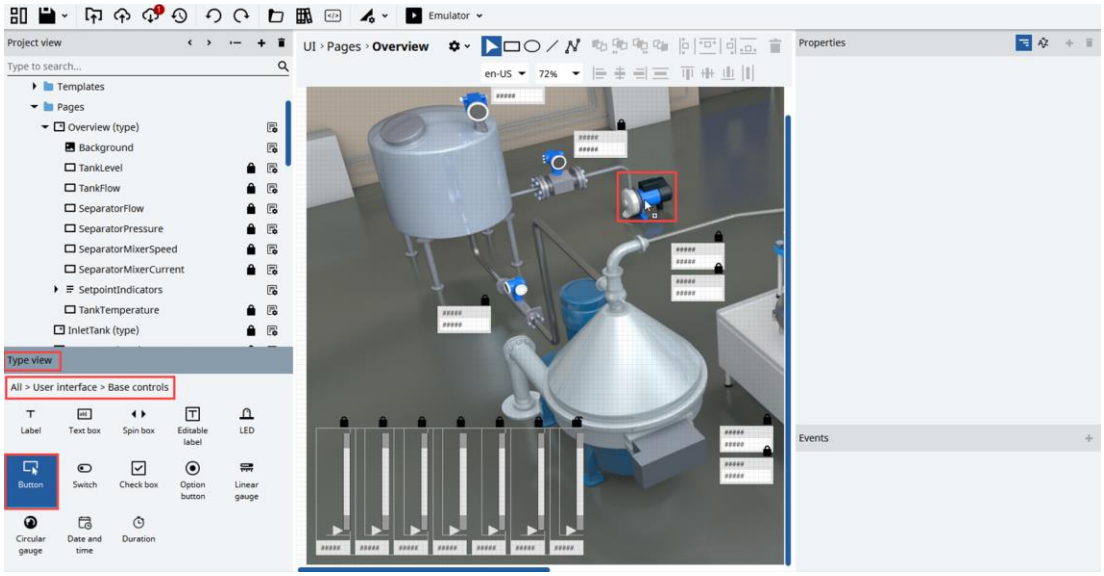
> You can also drag and drop the II101 variable to the AnalogVariablePointer field as you did on the AnalogItem widget, it will lead to same result



Display popup

1. Add a transparent button to open the inlet pump popup.

   Select the **Overview** display in the **Project View**. From the **Type view** *browse* to User **Interface > Base controls** and *drag* a **Button** onto the inlet pump.

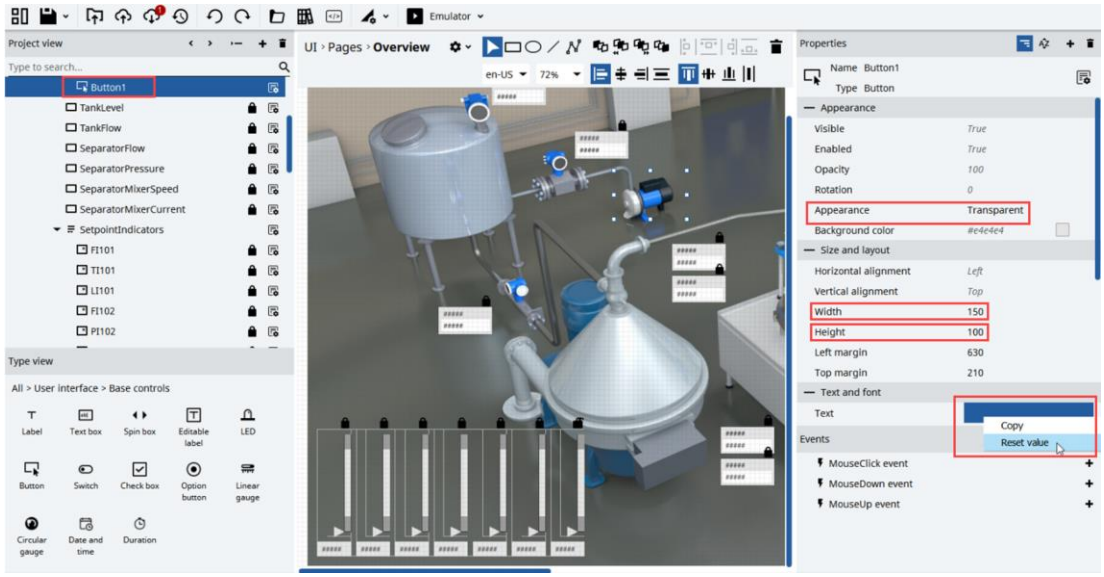


2. Set the button to transparent.

   Click on the **button** and edit its **Properties** in the right column, set:

   - **Appearance:** ***Transparent***
   - **Width:** ***150***
   - **Height:** ***100***

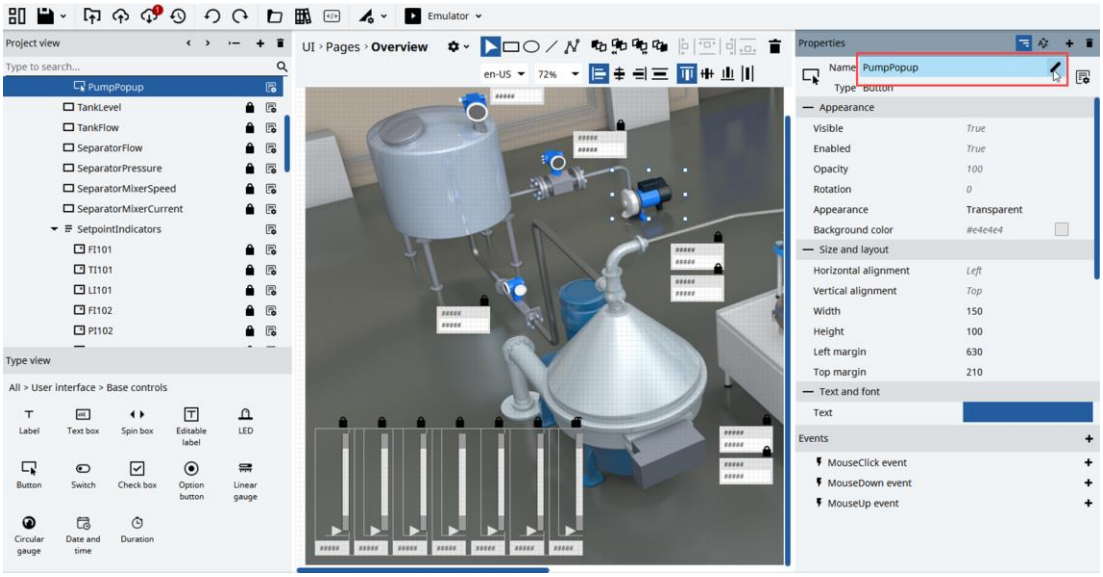
   Clear the text by *right clicking* on the **value** and select ***Reset value***.



3. Rename new button

Click on the **pencil icon** in the **Name** parameter to rename new button ***PumpPopup***.

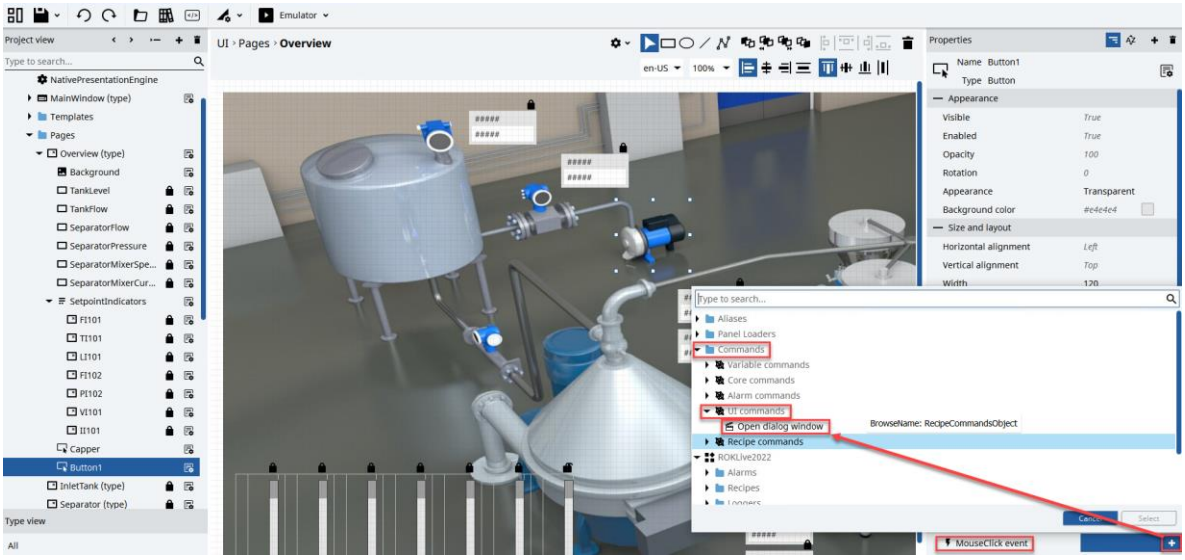
> Renaming is also available by right clicking any item in its dropdown menu.



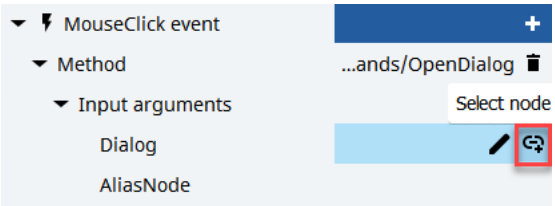
4. Configure the **MouseClick event**

Now you will configure the button to open a Popup when clicked, *click* the **+** icon at the bottom of the Button next to MouseClick event and *browse* to **Commands > UI Commands > Open dialog window** and *click* ***Select***.

> Multiple events are provided such as set variables value, toggle variables, interact with graphical objects and much more. A single command (such as button) can trigger multiple events by using the "+" icon and adding a new function



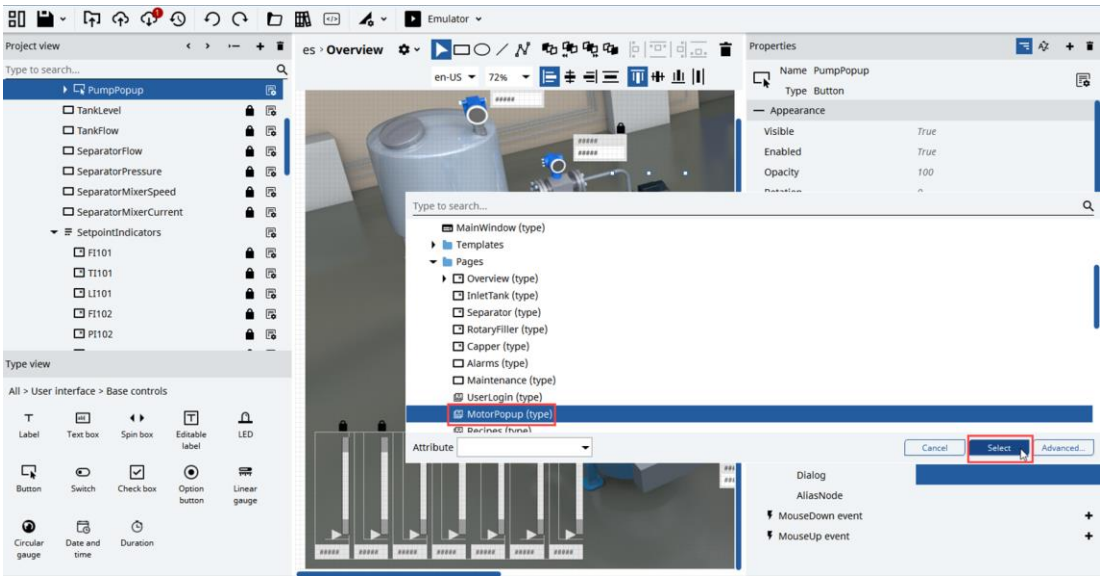
5. *Click* on the ***Dynamic Link/Select Node*** button.



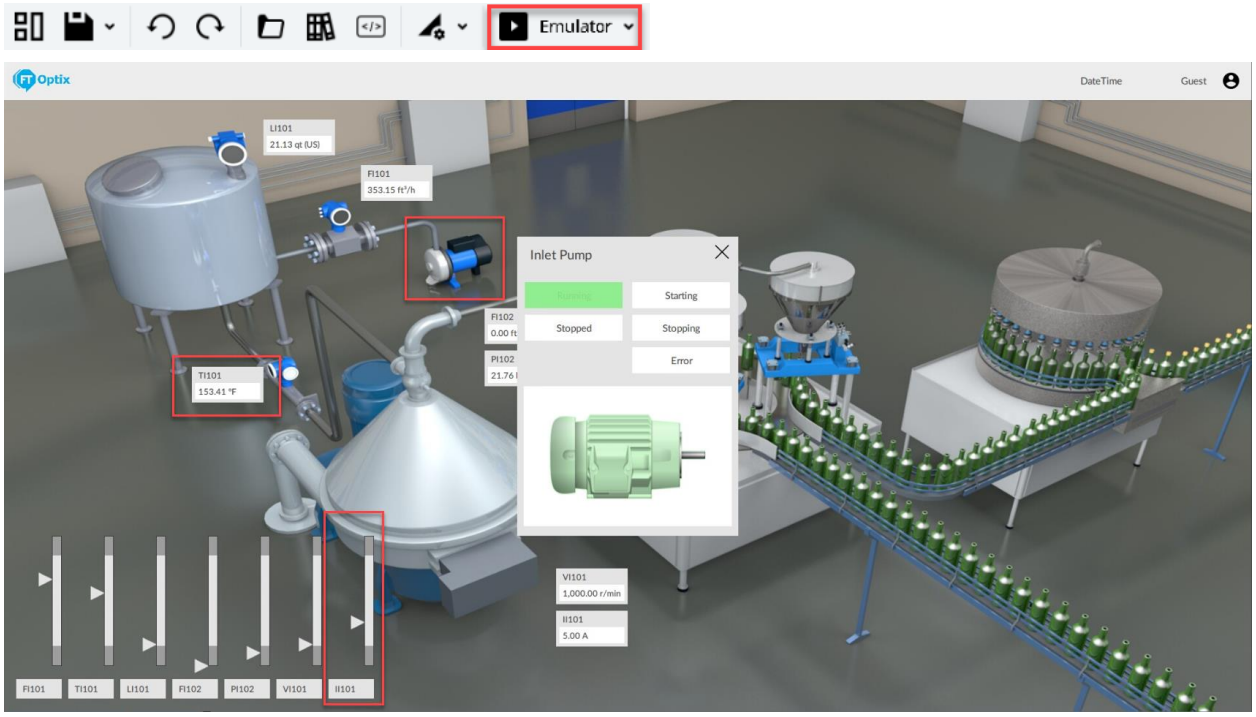
6. Select which popup to show.

   Use the project browser to point to the **MotorPopup**. There is no need to specify any AliasNode here as you are not passing any argument to the window, as you did when using the AnalogItem widget.

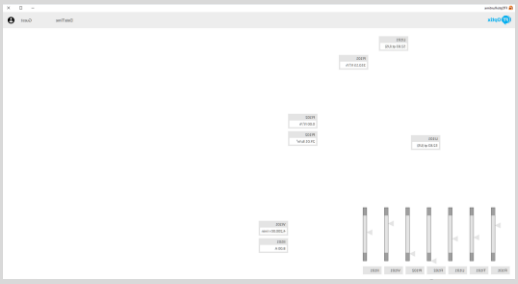
> The Aliases concept is widely used in FactoryTalk Optix™, the entire IDE is based on object-oriented programming, you first define a Type (e.g. a widget) and create Instances of that item passing it different arguments around the application.



6. Now run the **Emulator**, you will see the AnalogInput and the SetPoint Indicator widget animated. *Click* on the **inlet pump** and the inlet popup display will open.



When the OverView screen opens and it looks like the screen below, without a background image.

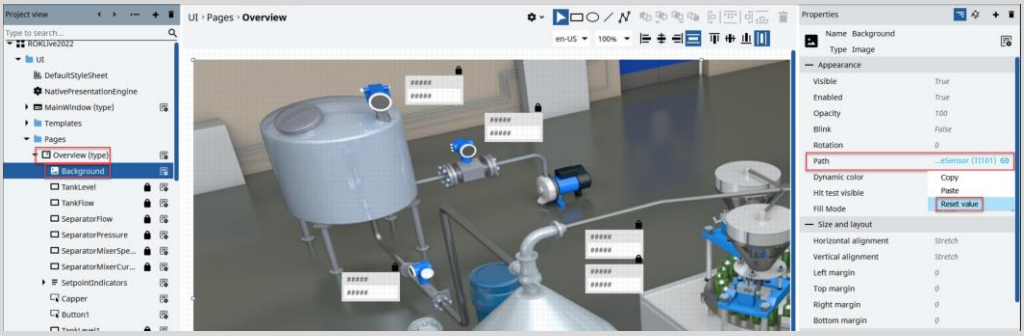


By dragging the variable to the objects on the screen, you dragged and dropped the variable accidently onto the background and not the object.

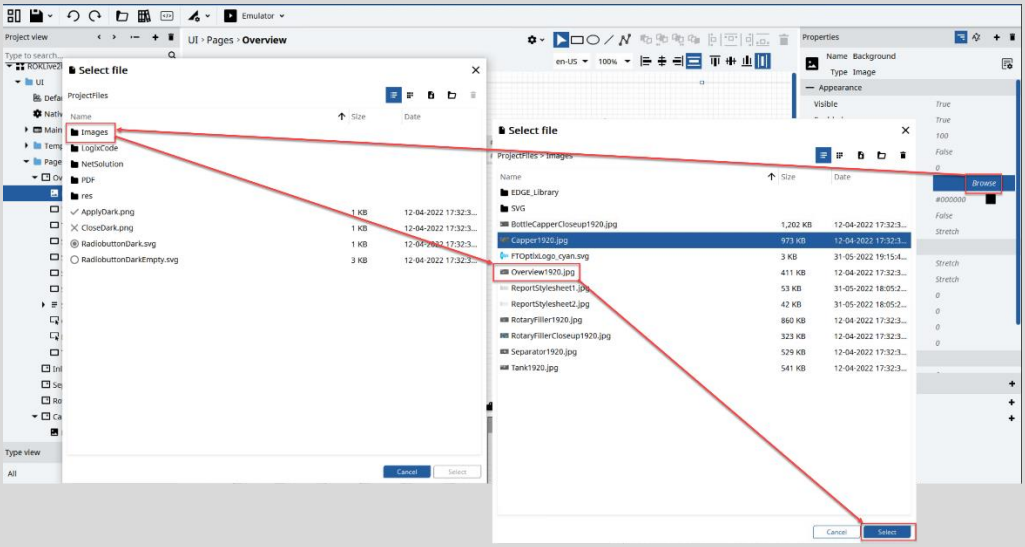
In effect you have set the browse path to the tag name.

These are the steps to get your background image back.

Go to the **Background** object on the **OverView** screen, in the properties, locate the **Path**, *Right Click* on the **TT101 or TI101** tag and select **Reset Value**.



Now *Browse* to the **OverView1920.JPG** file in the **Images** folder.



7. Close the emulator when you have finished testing.

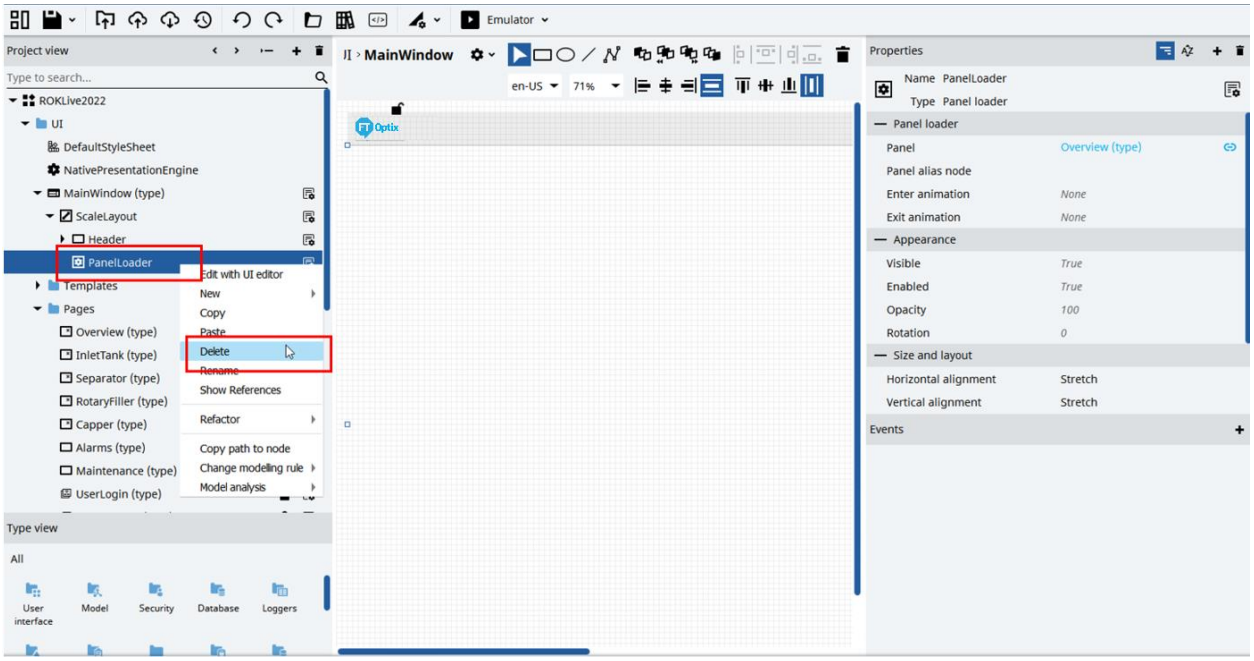
Display Containers

FactoryTalk Optix™ applications can be navigated using two solutions:

- Panel loader: a page display that can be navigated using buttons or UI commands, browsing can be done with buttons, scripting or events, at design time or run time
- Navigation panel: a tab-style navigation menu, all pages are added at design time and browsing is done manually interacting with top bar

1. Adding a NavigationPanel to manage multiple pages.

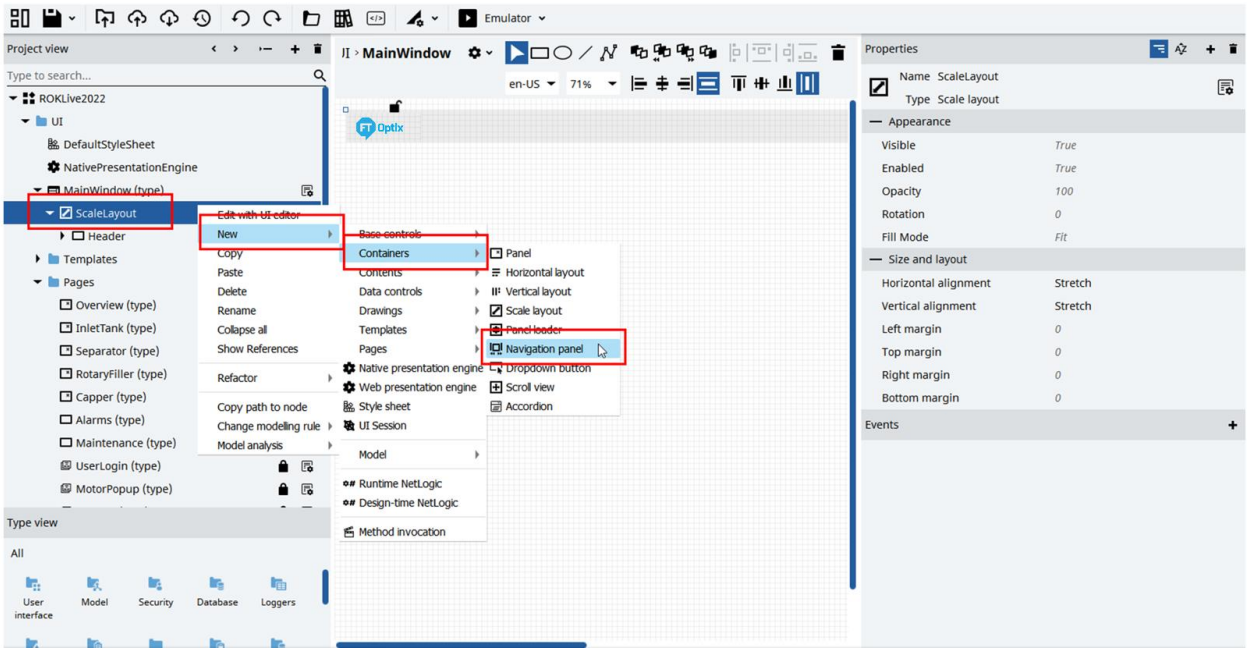
   Browse to **MainWindow**, expand the project tree and *right click* on **PanelLoader** to delete it.



2. Adding a NavigationPanel.

   Right click on the **ScaleLayout** and add a **NavigationPanel** to the project.

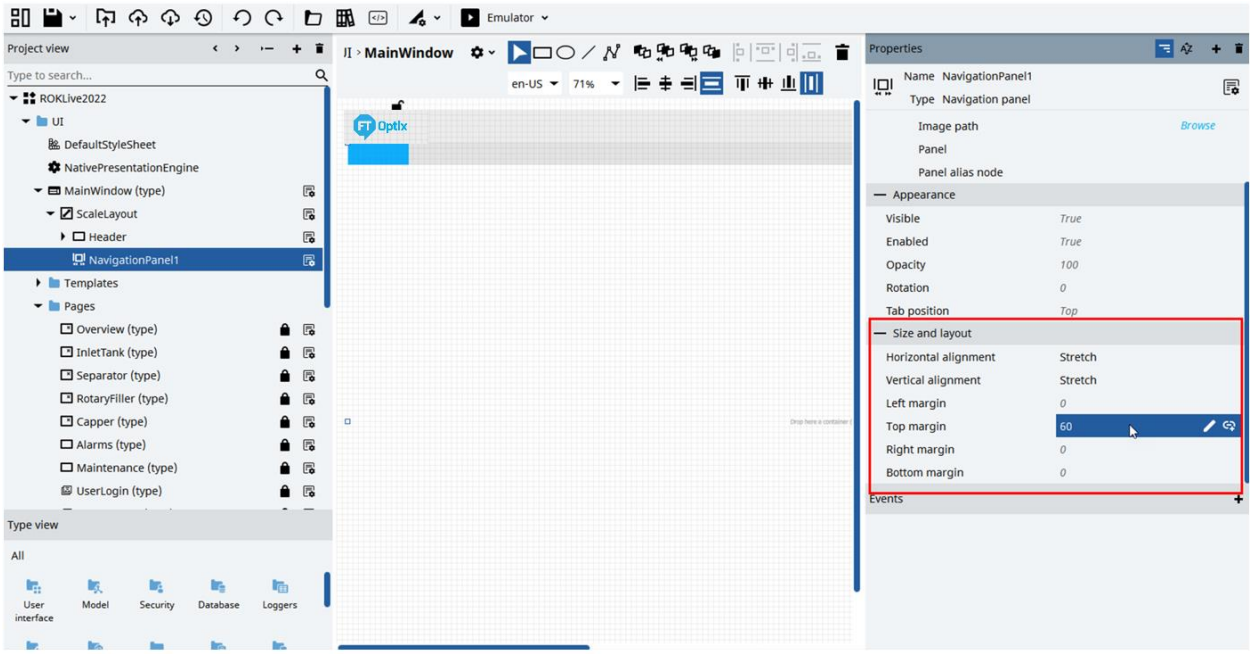
ScaleLayout is a specific container used in a project to resize automatically its children maintaining objects proportions, this is a very useful container to use for deploying projects on HMIs with different screen sizes without user modifications.



3. Adjusting the **NavigationPanel** appearance

   Set the **NavigationPanel** properties to:

   - **Horizontal alignment:** ***stretch***
   - **Vertical alignment:** ***stretch***
   - **Left margin:** ***0px***
   - **Top margin:** ***60px***
   - **Bottom margin:** ***0px***
   - **Right margin:** ***0px***

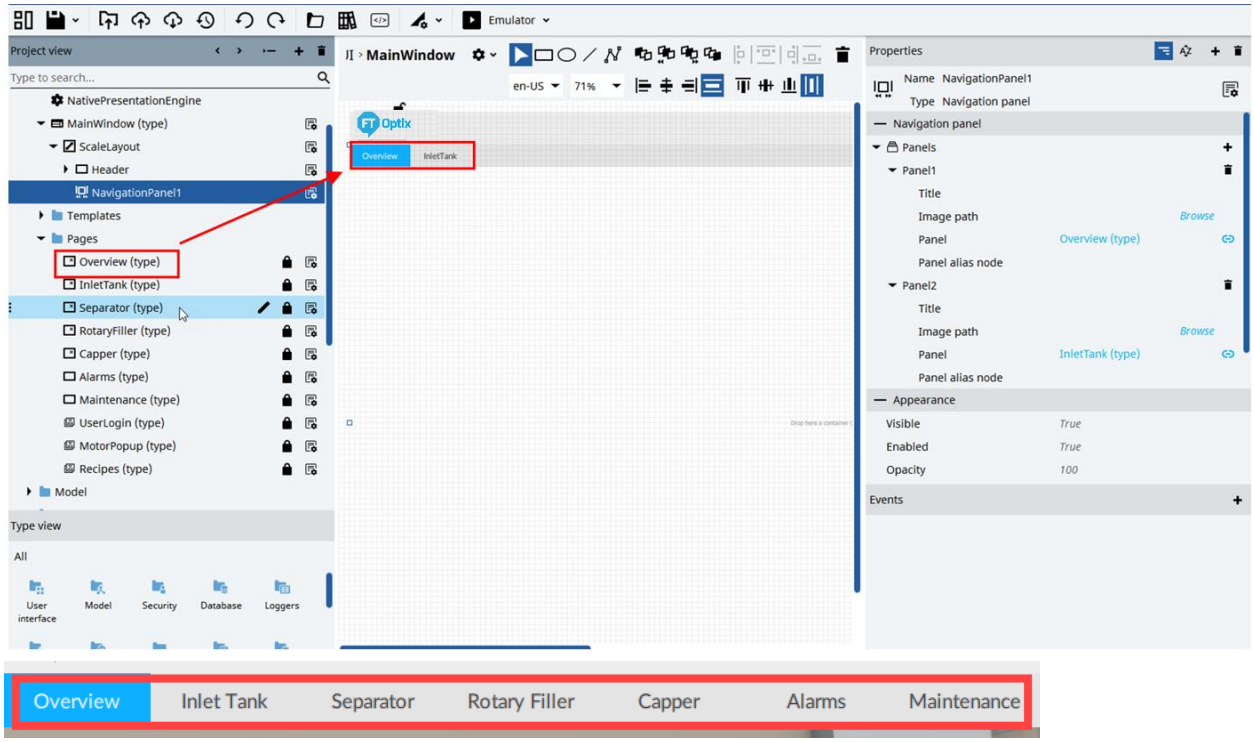


4. Adding pages to the navigation menu.

   Adding pages to a Navigation Panel is a straightforward process, it can be done in two ways:

   - Adding pages to the Panels properties on the right column
   - Dragging pages to the navigation bar

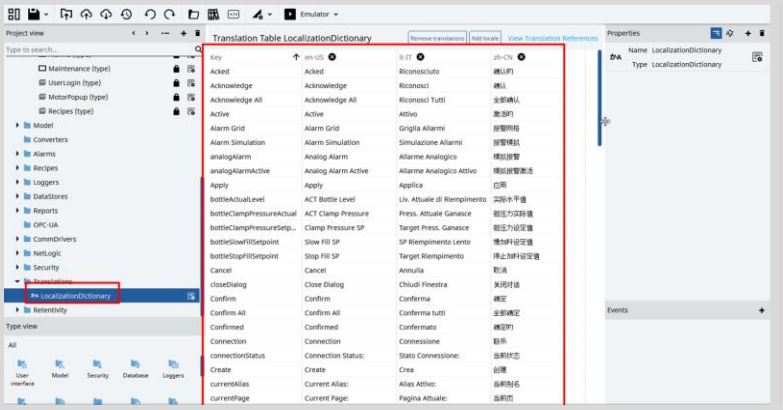
   Expand the project tree and start dragging pages from the left column to the grey bar of the Navigation panel, you will see the tabs being added both in the top bar and in the property's column in the right. Repeat the process for every page of the project (excluding the three popups).



5. Managing tabs translations.

Every tab can be customized with localized text or icons using the properties column

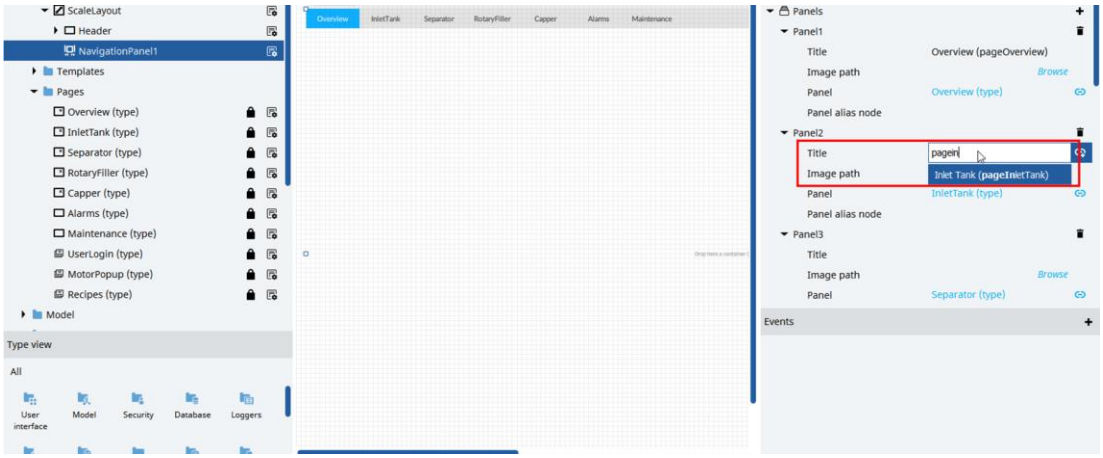
Translations are stored in a specific table, each line corresponds to a translation key, for each key there will be a column per language



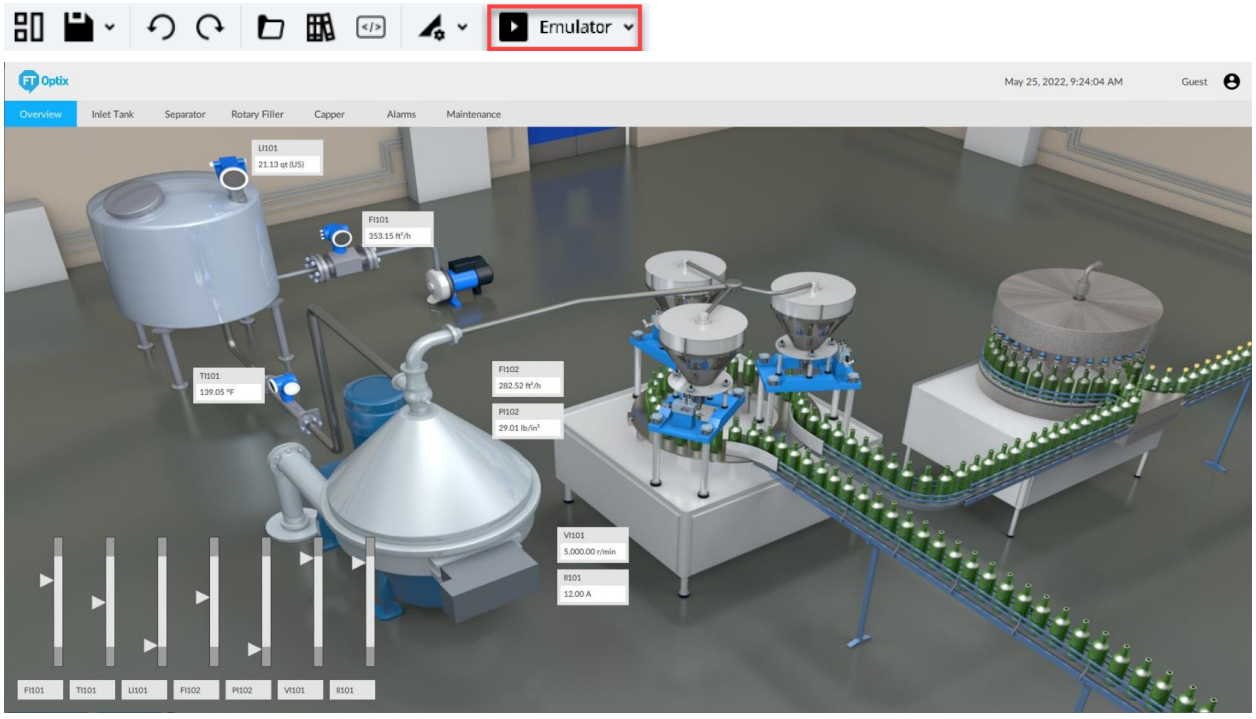
For each page you will add the translation keys in the **Title** property, start typing the translation key, a dropdown menu will appear, choose the correct key with **keyboard arrow** and hit **Enter** to confirm. For this project you will use:

- **Overview:** ***pageOverview***
- **InLetTank:** ***pageInletTank***
- **Separator:** ***pageSeparator***
- **RotaryFiller:** ***pageRotaryFiller***
- **Capper:** ***pageCapper***
- **Alarms:** ***pageAlarms***
- **Maintenance:** ***pageMaintenance***

Translations have already been added with English, Italian and Chinese for each key



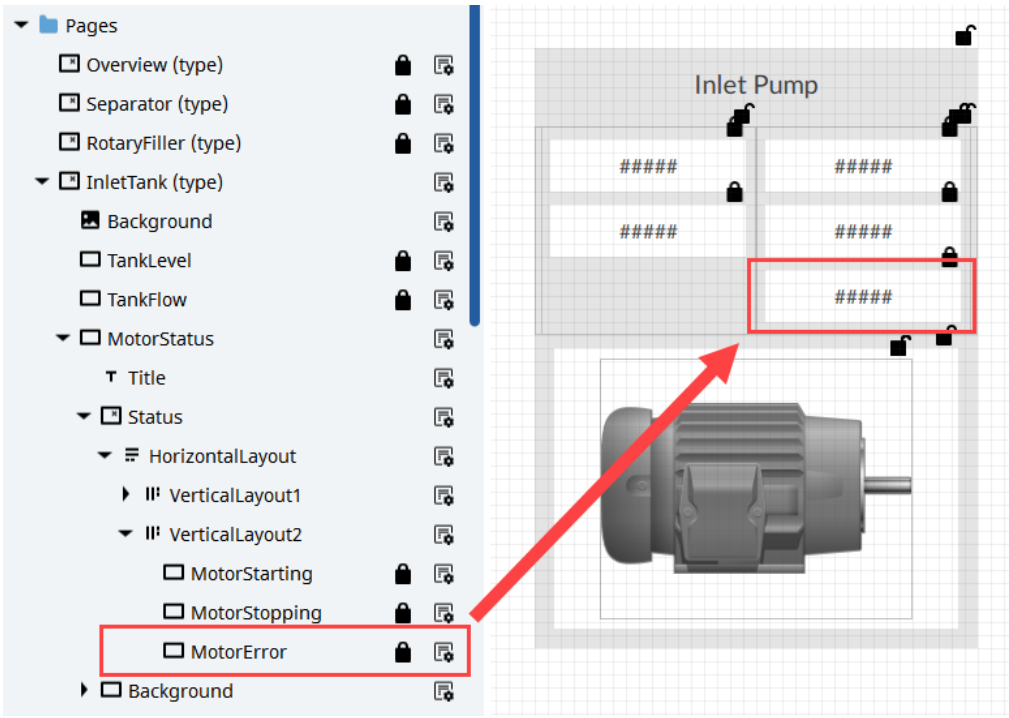
6. Now run the emulator, and you will see the Navigation Container you have just created.



7. Close the **emulator** when you have finished testing.

Objects, Types, Instances, Aliases, SVG Object Color change and Key Vale Converters

In this section, you will add a Template Object (folder “Templates”) to the display **Inlet Tank**. The template object is called ***SingleStatusRed(Type)***. In particular, the instance name of SingleStatusRed(Type) will be ***MotorError.*** This state represents the motor error condition



This will be the result:

Read about FactoryTalk Optix™ Objects

An object is an element that identifies a physical (a motor) or non-physical (a digital alarm) object in the project.

An object has the following characteristics:

can contain variables

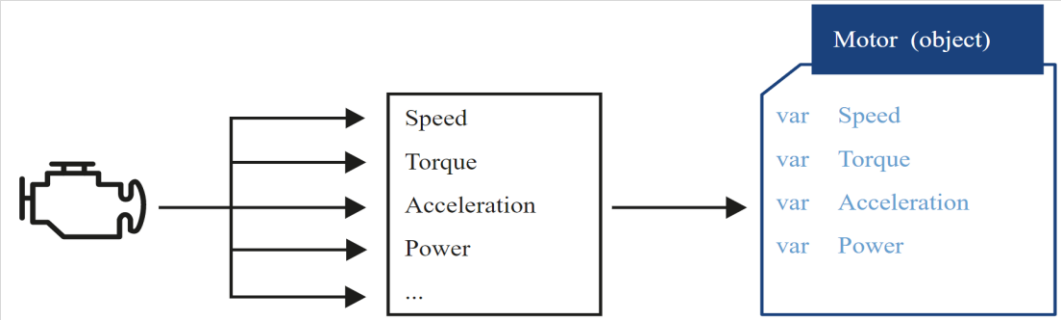
can contain other nested objects (for example a motor can contain objects that represent its components)

has attributes that describe it

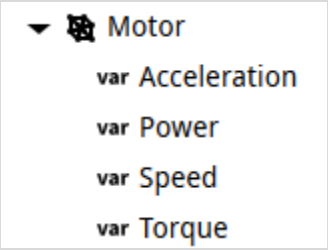
Example

In the real world, a motor is a mechanical object with specific operational characteristics, e.g. speed, torque, acceleration and power.

In FactoryTalk Optix™*, the motor just described is represented with an object (Motor) which contains the variables corresponding to its functional characteristics (Speed, Torque, Acceleration, Power).*



Below is an example of how the same object is structured and displayed in FactoryTalk Optix™ Studio:



Attributes

An object always has the following attributes:

Attribute	*Description*
BrowseName	*Name of the object in the project*
Description	*Description of the object*
Display name	*Name of the object, translatable and usually used to be displayed in the interface at runtime*
NodeId	*Unique identifier, assigned automatically*

Object types and instances

An ObjectType is a "model" object from which similar objects will be derived, called instances. However, instances can contain other specific objects and/or variables.

Creating instances is advantageous in terms of speed and consistency, as any changes made to the type are automatically made to the instances.

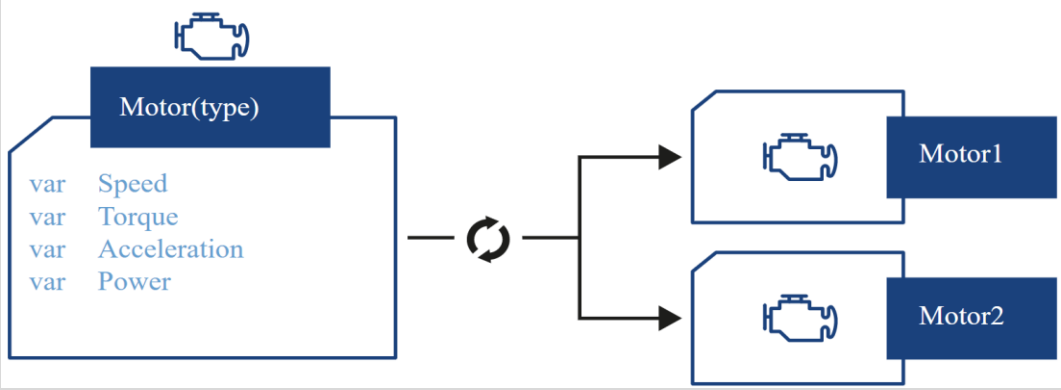
Important

if an object type is deleted, the instances are also deleted.

Example

Let us consider that there is a machine consisting of two identical motors.

Instead of configuring individual objects for different motors, in FactoryTalk Optix™ it is possible to configure the object type ***Motor*** *with all its functional characteristics (the* ***Speed****,* ***Torque****,* ***Acceleration*** *and* ***Power*** *variables) and use it to derive multiple instances, in the example* ***Motor1*** *and* ***Motor2****.*



Read about FactoryTalk Optix™ Information Model

Nodes and information model

In FactoryTalk Optix™, all the objects, object types, variables, variable types and methods are generically defined as nodes.

The concept of information model indicates the set of nodes that describes the structure and characteristics of a node in the project. Each object, variable, type or method is therefore described by its own information model.

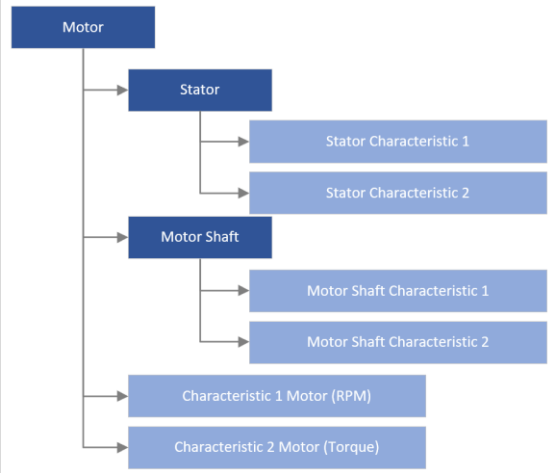
Example

A motor consisting of two parts, a stator and a crankshaft, can be represented by the following information model. All the elements described are nodes, in particular objects (in blue) and variables (in pink).

Information Model of a Project

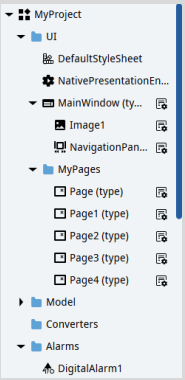
The set of all objects, variables, types and methods in a FactoryTalk Optix™ project is conventionally defined project information model.

Example



*Below is an example of how the different nodes of a project are displayed in FactoryTalk Optix™ Studio. The project information model is the information model of the **MyProject** root node.*

The nodes are organized in folders for better project readability.

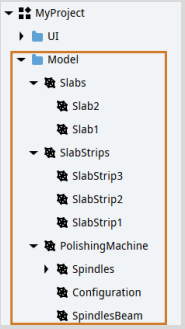


Information models to represent physical objects

Generally, it is recommended to structure the project to mirror as much as possible the machine structure for which the application is created; the project is thus more readable. Also, in the case of applications for modular machines, it is convenient to represent these modules by structuring them similar to the physical world; this is to facilitate their reuse in the project.

Example

Below is a partial example of a project in which the structure of the objects mirrors the structure of the machine:



Read about Alias

Introduction

*An alias is a particular variable type that points to a source node indicated by its unique identifier (**NodeId** attribute). It links two areas of the project information model and makes it possible to set up dynamic links to create particular dynamic logics that would otherwise be impossible.*

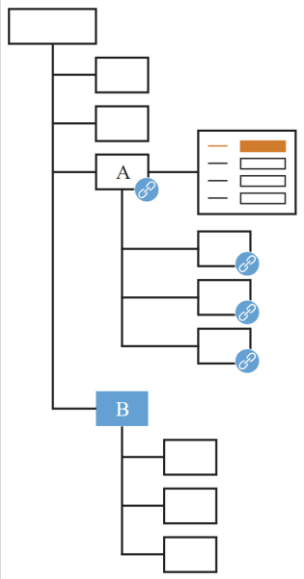
An alias is created in an object, or type of object, generically defined as parent node. The source node it points to identifies the source area, which includes the same source node and all its child nodes.

Note

Typically aliases are created in graphical objects or in folders containing graphical objects, to create dynamic user interfaces (for example, when creating a widget that represents a motor type and that dynamically shows the values of different instances of the same type).

A short overview of how an alias works

*This diagram summarizes how aliases work. In object **A** there is an alias that points to the source node **B**. From node **A** and/or any of its child nodes, dynamic links can be created with node **B** and/or any of its child nodes.*

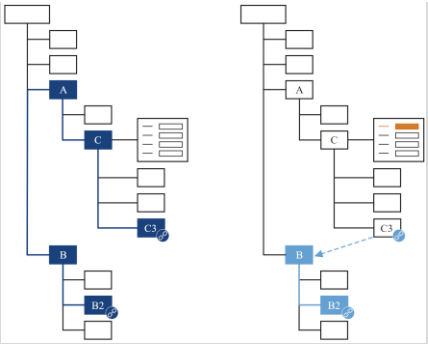


Dynamic links between aliases

When an alias is not exploited, a dynamic link is created via a relative reference/path from a parent node to the source node.

*When an alias is present, stronger dynamic links can be created. In fact, thanks to its reference to the source node unique ID (**NodeId** attribute), the alias acts as an "intermediary" between the parent node and source node,*

*Below is an example where it is wanted to set the value of node **C3** using the dynamic link to node **B2**. In the first case a dynamic link is used and the reference is ../../B/B2. In the second case, thanks to the alias set on node **C**, the link is set up exploiting the alias and the reference is {B}/B2.*



When to use aliases

Aliases need to be used to set up dynamic links in the following cases:

When the dynamic link source node, and therefore its child nodes, exist only at runtime (e.g. the session node). In this case, at design time a link is set up towards a node of the fictitious source area made available by the alias.

When the dynamic link source node, and therefore its child nodes, also exist at design time, but can change at runtime by effect of the set dynamic logics (e.g. in the case of widgets).

An alias is also useful as a "shortcut" for creating dynamic links to recurrent nodes of interest, without therefore having to look for the same source node every time. At design time, in fact, the source nodes are grouped and easily accessible in the dynamic links window, while at runtime it is sufficient to know the name of an alias to access the node to which it points.

Kind alias property

An alias always contains a Kind property, whose value is a reference node. More precisely, it is a reference to the object/variable type from which the source node derives, or, if the source node is not derived from a type, a reference to the same source node. The reference node is required to describe the source node information model. In fact, the alias displays the source area nodes to the parent node and to all its child nodes based on a known information model, which corresponds to the Kind node model.

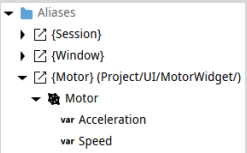
When the source node of an alias is set, FactoryTalk Optix™ Studio *automatically assigns a value to the Kind property. For example, if the source node is a Motor instance, the Kind property points to the Motor node.*

However, when the source node of an alias is set at runtime only, e.g. in the case of widgets, the Kind node must be specified. Thus links can be created in the dynamic links window by exploiting the nodes of the Kind node information model, which at design time represent the actual project nodes at runtime.

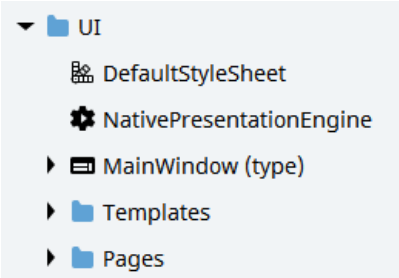
How aliases appear in the dynamic links window

In FactoryTalk Optix™ Studio, in the dynamic links window, the aliases are grouped in the Alias folder, except for the aliases for Dynamic panel objects generated automatically by FactoryTalk Optix™ Studio, which are grouped in a specific Dynamic panels folder.

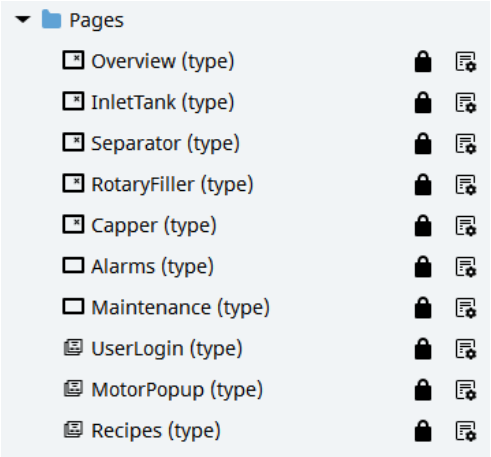
In the window, each alias is identified by a node with icon  and its name in curly brackets. The node can be expanded to navigate the information model of the node to which the alias points, and to select the source node of the dynamic link.



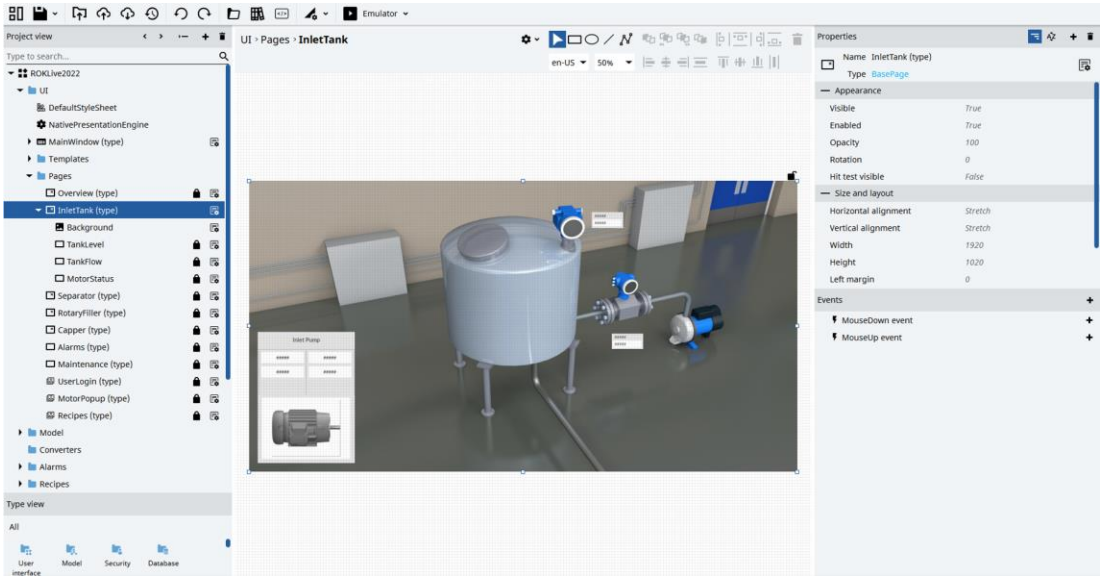
1. *Open* the folder/Node ***UI***



2. *Open* the folder/Node ***Pages***

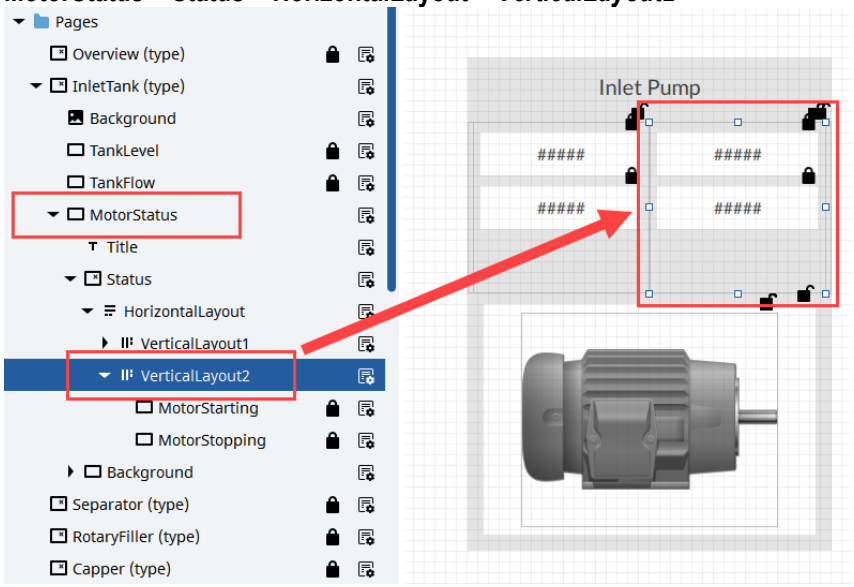


3. *Double click* to *open* and unlock ***InletTank(Type)***. The view will be as follows:

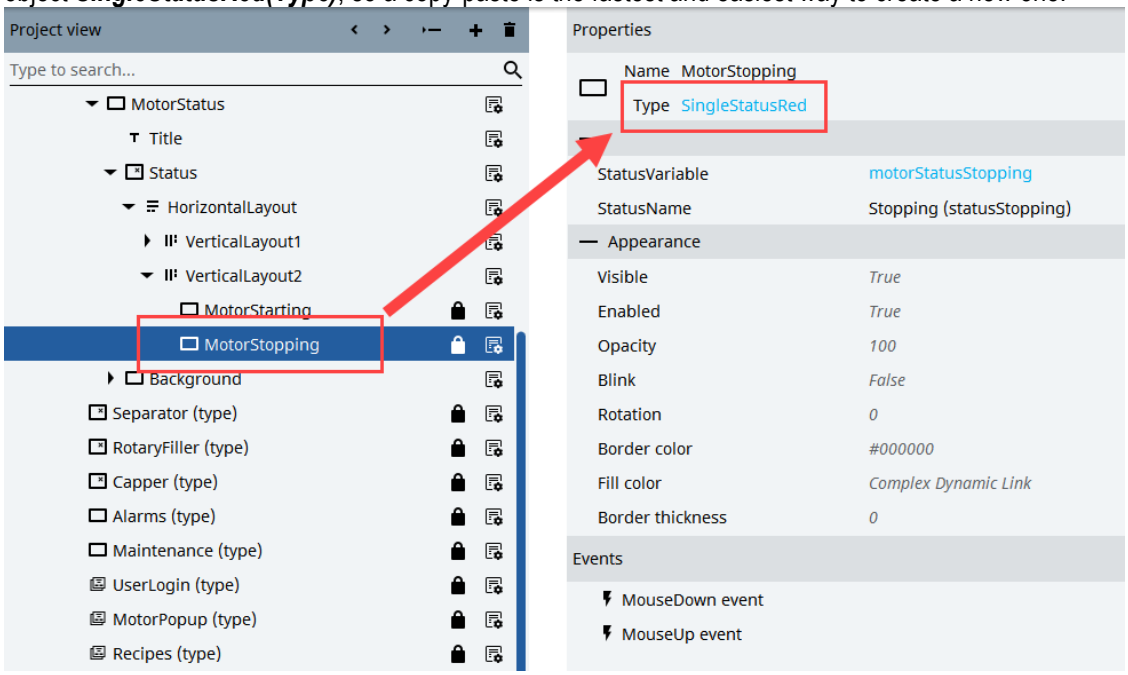


4. *Open* the panel ***MotorStatus*****:** *double click* to open and unlock. Then proceed according to the following path:

MotorStatus > Status > HorizontalLayout > *VerticalLayout2*

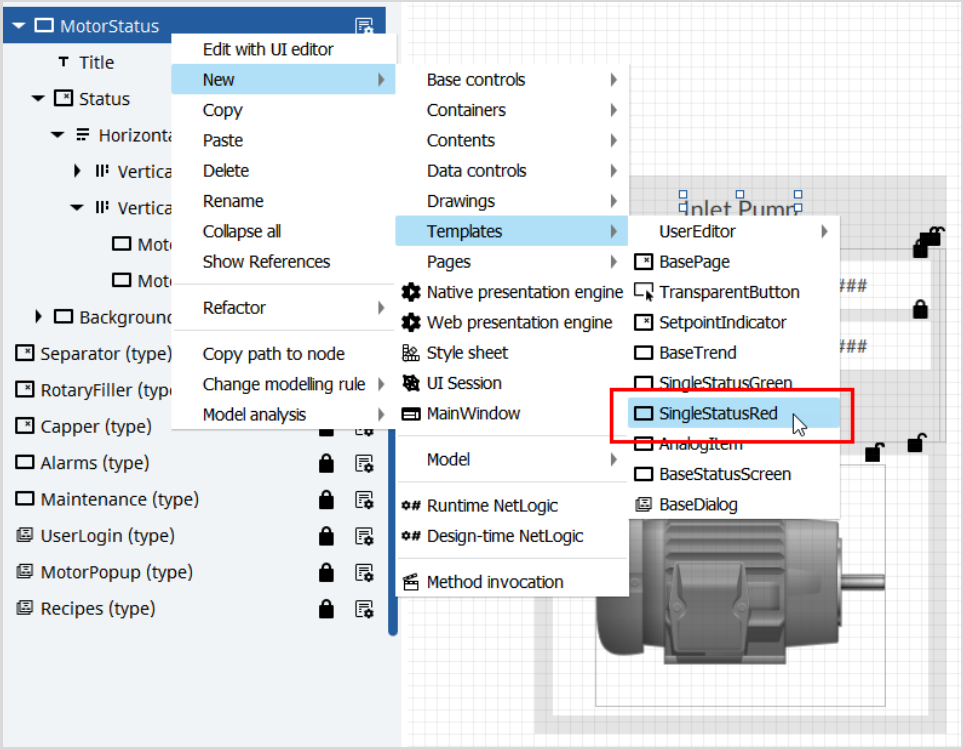


5. By selecting the ***MotorStopping*** object, in the Properties panel you can see that this is also an instance of the object ***SingleStatusRed(Type)***, so a copy-paste is the fastest and easiest way to create a new one.



Instantiate a new object – *The alternative would be to right click on the container object (**VerticalLayout2** in this case) and then instantiate a new **SingleStatusRed(Type)** object following this path:*

New > Templates > SingleStatusRed

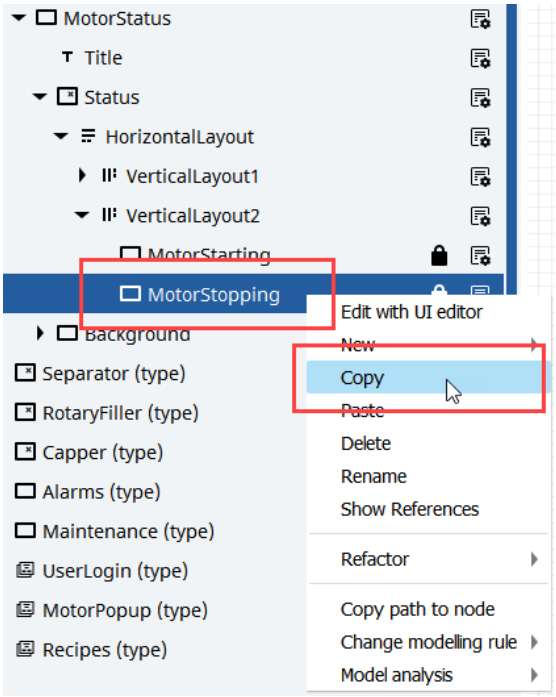


At this point, however, the alignments will have to be done manually.

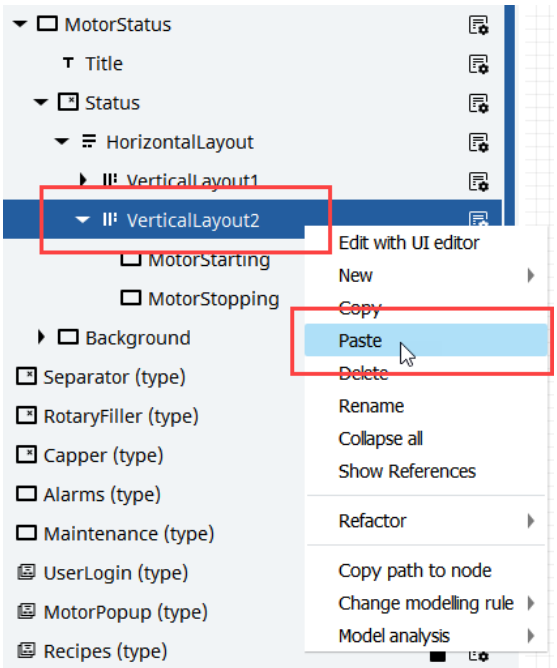
6. *Copy-Paste* the object ***MotorStopping*** with the keyboard shortcut CTRL+C / CTRL+V

   Or by following these steps

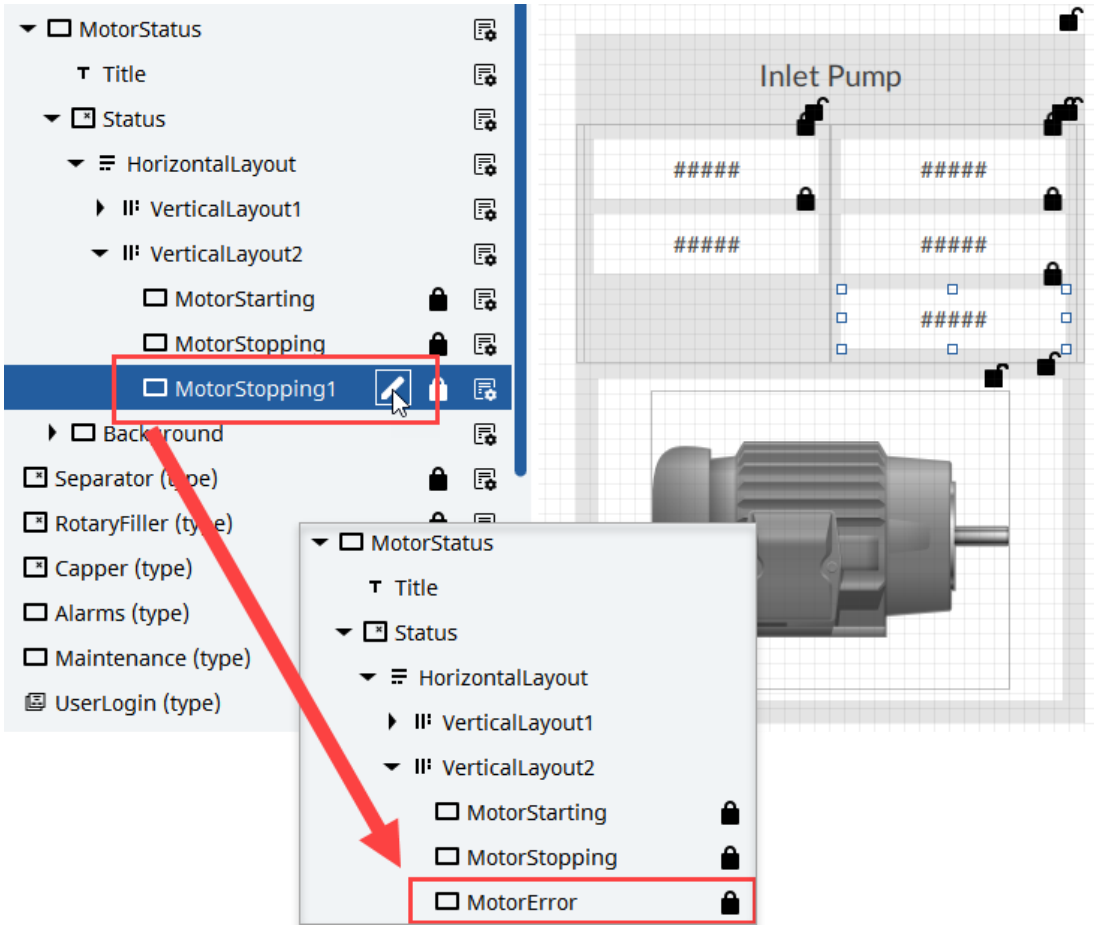
   Right click on ***MotorStopping*** then *select* ***Copy***.



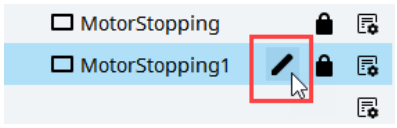
   Right click on ***VerticalLayout2*** (the desired place for the new object) then *select* ***Paste***.



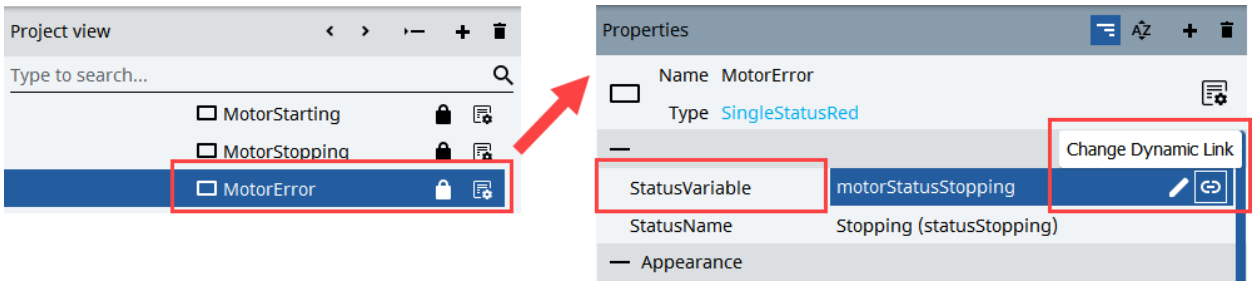
7. *Rename* ***MotorStopping1*** to ***MotorError***: this object will show the InletTank motor error status.



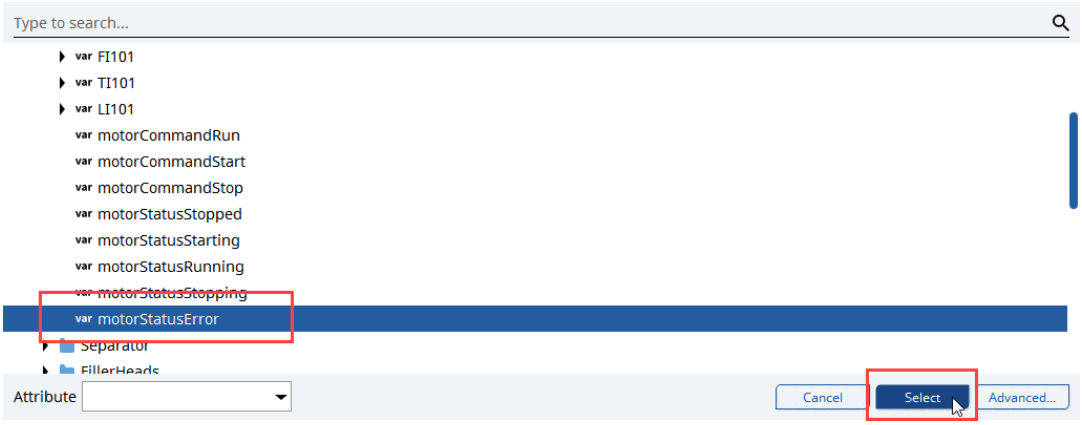
Select the ***pencil icon*** (modify) to rename an object.



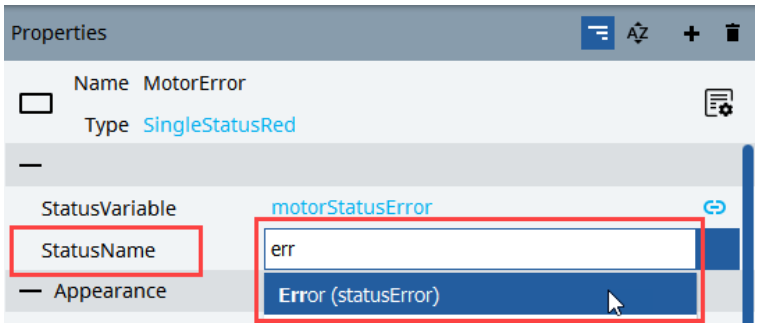
8. Maintain the Project view selection on ***MotorError***. To modify the variable bound to **MotorError *StatusVariable*,** *select* the ***chain icon*** on the Properties panel**.** The DynamicLink Popup will be opened.



9. *Select* the new variable to bind ***motorStatusError*** on the DynamicLink Popup.

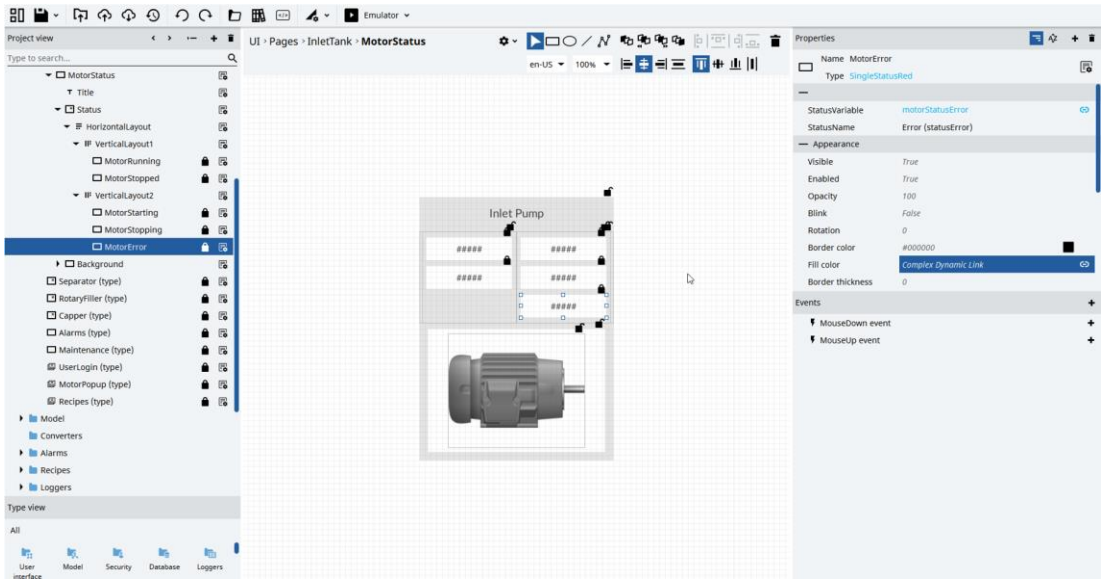


10. Keep the Project view selection on **MotorError**. To modify the variable bound to **MotorError *StatusName***, *select* the pencil icon on the Properties panel**.** Start to *type in* ***Error.*** When the translation appears under the entry field, *press the Enter key* or *double click* the desired translation suggestion.



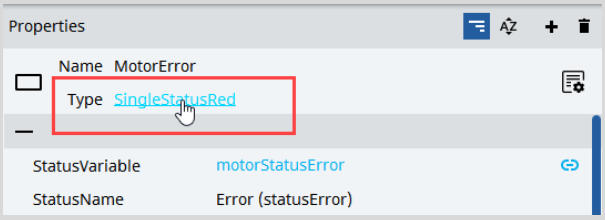
> *Tip* ***Translations*** *- When the desired translation is already present in the* ***Translation Table LocalizationDictionary****, just start typing the beginning of the word to see the suggestion for compatible translations you can enter.*

11. The configuration of the new MotorError object is completed.



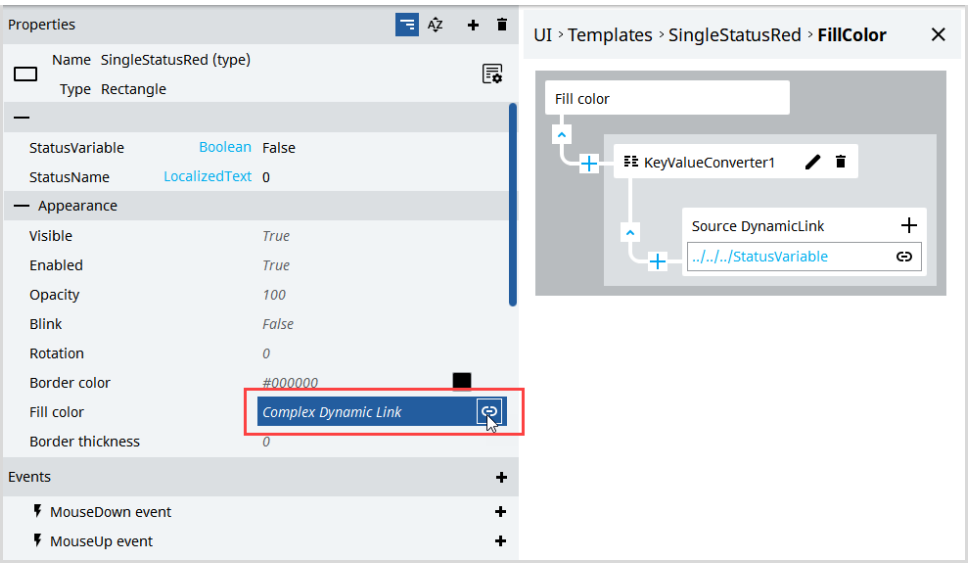
Read about Type definition

MotorError *is an instance of* ***SingleStatusRed(Type)****. The Type definition can be viewed by selecting the light blue link* ***SingleStatusRed*** *of the instance Properties panel.*



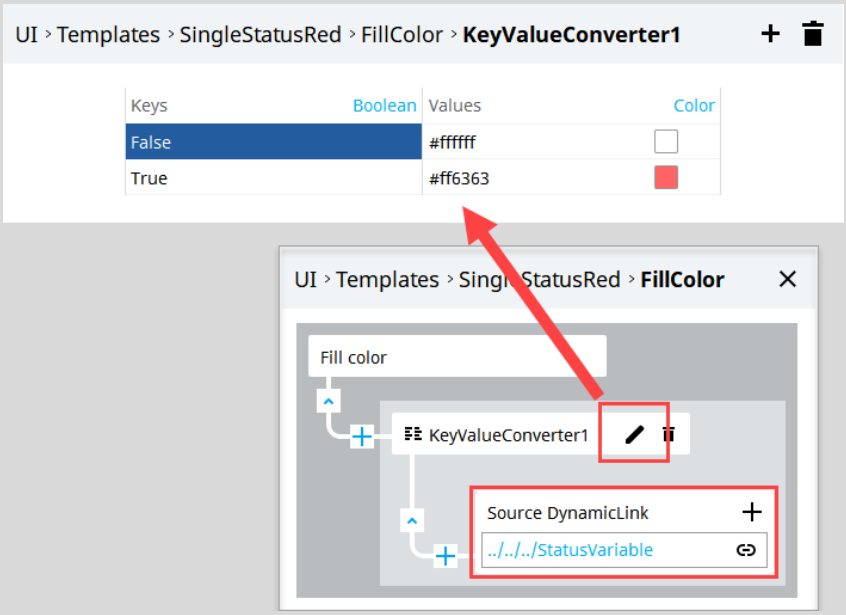
Advanced Property View

The colorization logic of the background is made with a ***KeyValue Converter.*** *To see how it's built, select the* ***chain icon*** *next to* ***Complex Dynamic Link****. The Advanced Property View will appear on the right of the screen.*



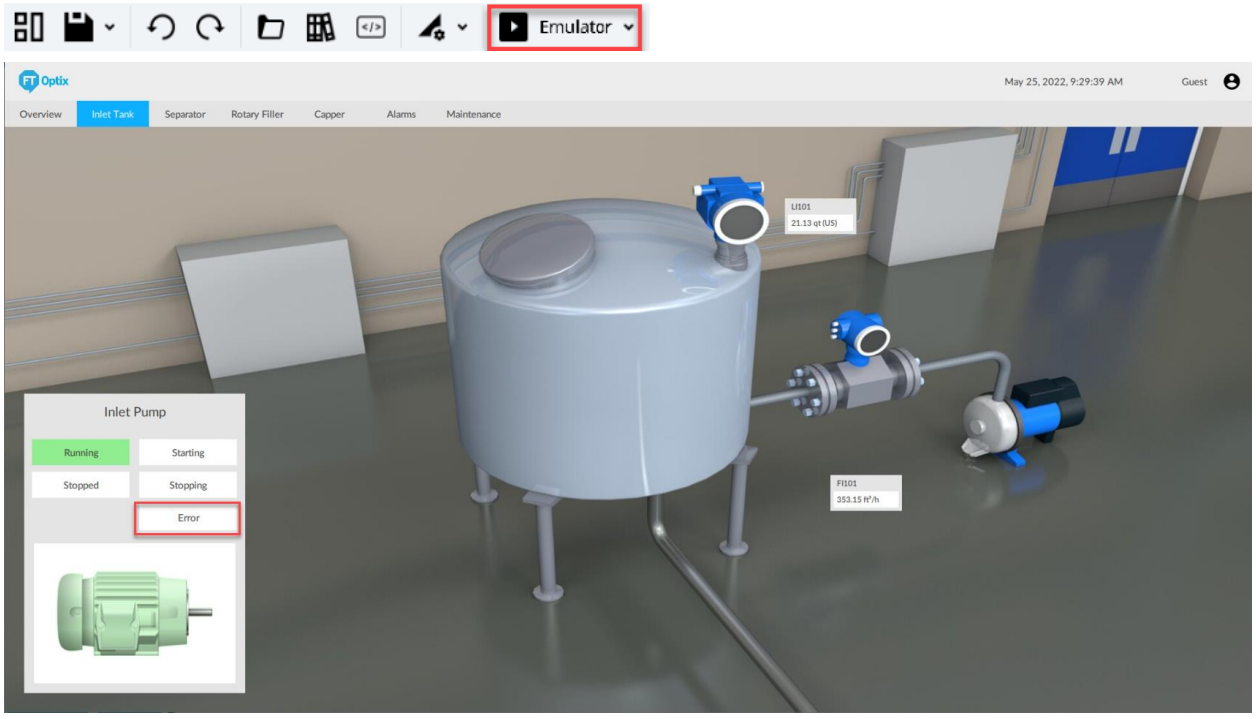
Read about Colorization Logic

*The colorization will change based on the **Source DynamicLink** variable **StatusVariable.** The corresponding status color will be determined based on the KeyValue Converter map. Select the **pencil icon** (modify) next to **KeyValueConverter1** to see the KeyValue Converter map.*



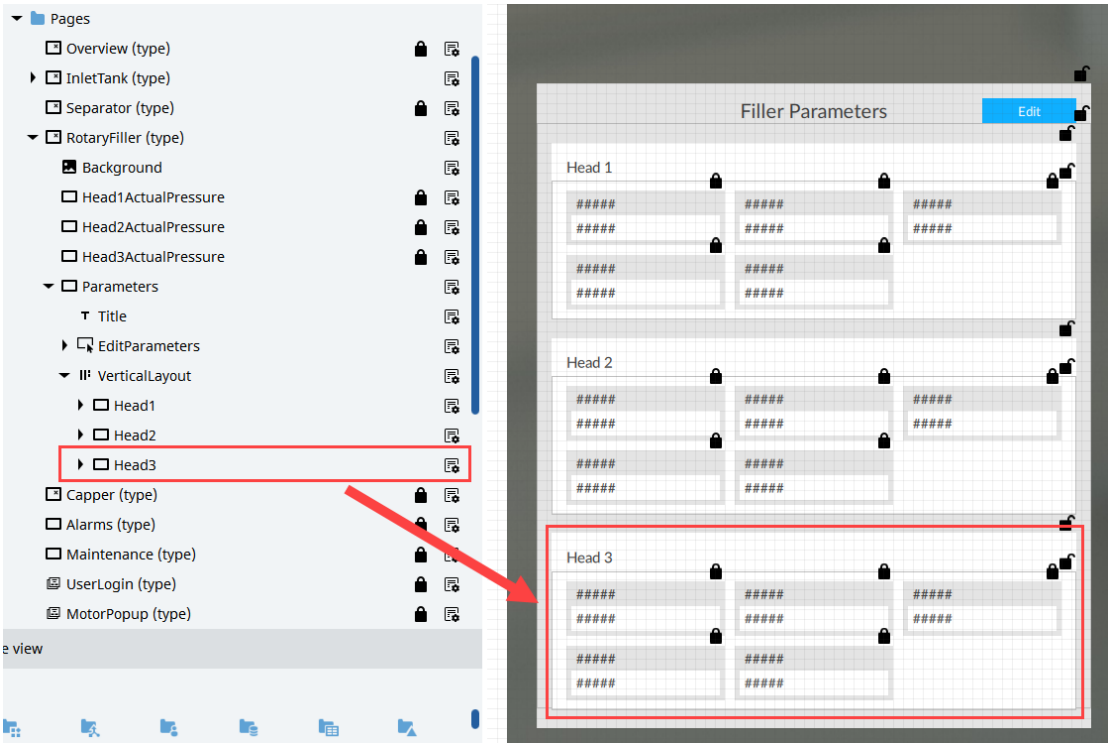
*Here, the background color of the rectangle will change to red when **StatusVariable** value will change to TRUE.*

12. Now run the **emulator** and click on the **Inlet Tank** tab.



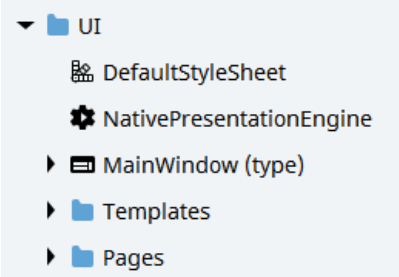
13. Close the **emulator** when you have finished testing.

Now you will add a Rectangle object containing five Template Object (folder “Templates”) to the Page **Rotary Filler**. The template object is called ***AnalogItem(Type)***. You will use the copy-paste capabilities of objects and the editing of DynamicLinks.

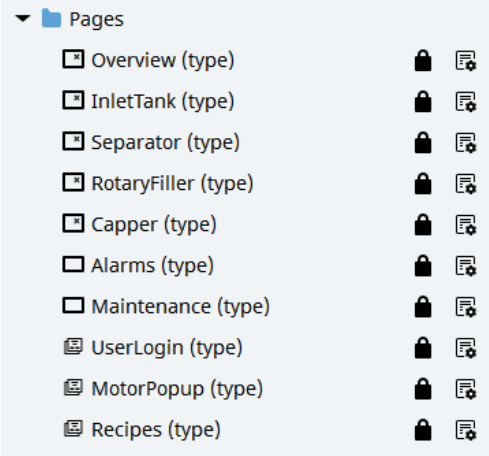


This will be the result:

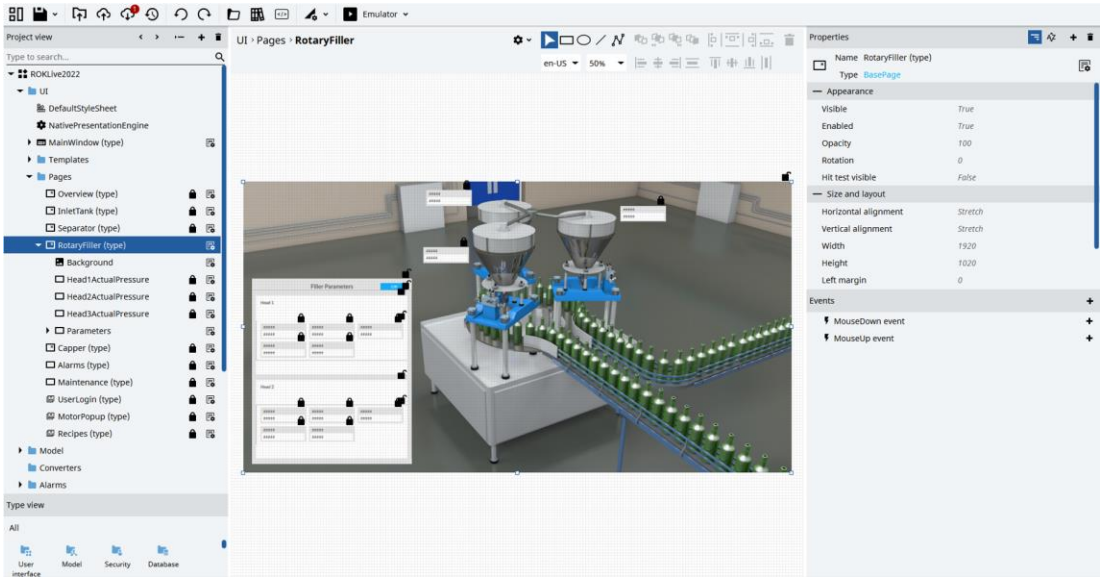
1. *Open* folder/Node ***UI***.



2. *Open* folder/Node ***Pages***.

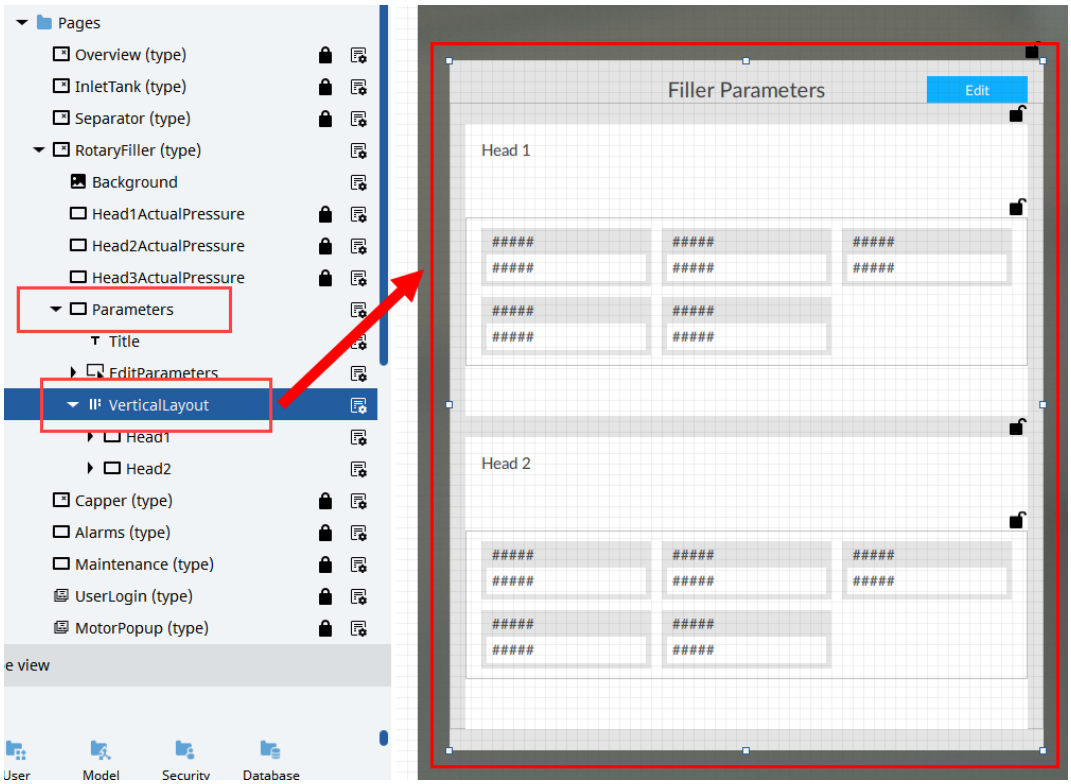


3. *Open* panel ***RotaryFiller(Type)*****:** *double click* to open and unlock. The view will be as follows:



4. *Open* panel ***Parameters***. Then proceed according to the following path:

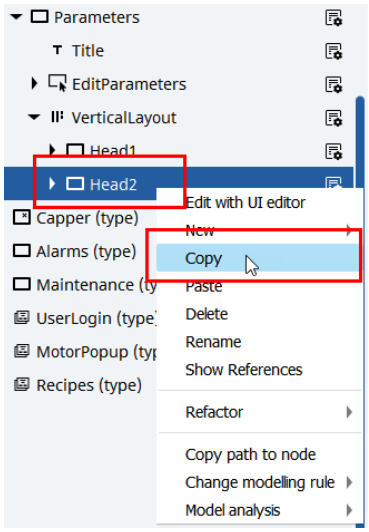
   Parameters > *VerticalLayout*.



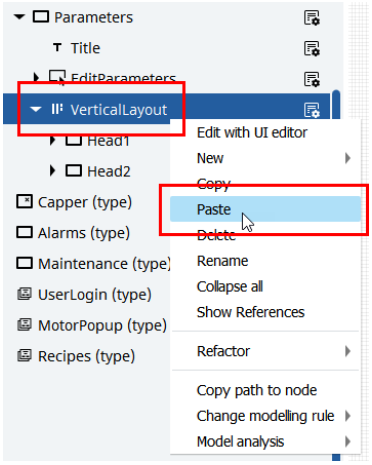
5. Considering that the new Head3 rectangle must be equal to ***Head1*** and ***Head2***, a copy-paste is the fastest and easiest way to create a new one. *Copy-Paste* the object ***Head2*** with the keyboard shortcut CTRL+C / CTRL+V

   OR following these passages:

   Right click on ***Head2*** then *select* ***Copy***.

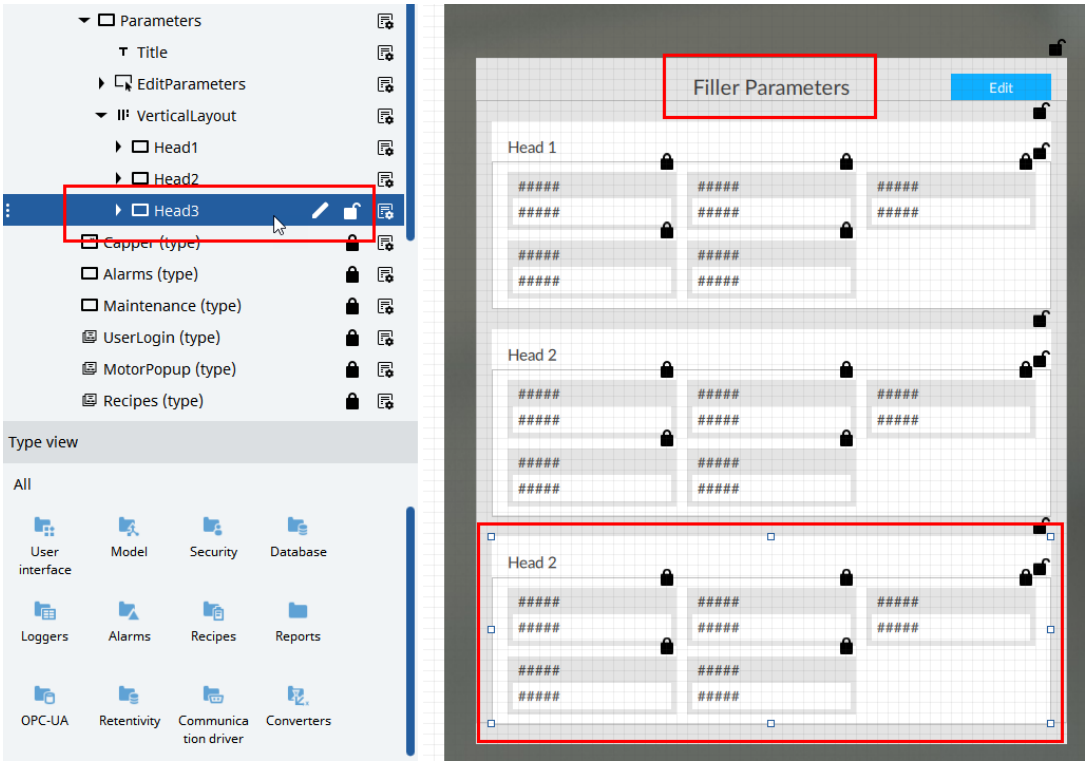


Right click on ***VerticalLayout*** (the desired place for the new object) then *select* ***Paste***.

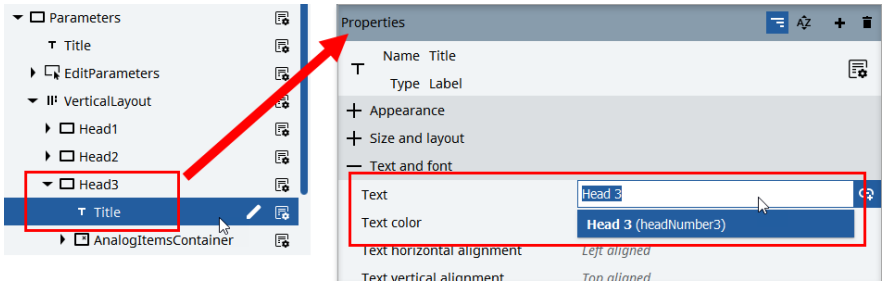


> *Tip* ***Vertical Layout / Horizontal Layout –*** *Copy-Paste of similar objects in a Vertical Layout / Horizontal Layout is particularly quick and easy because alignment and margin values of the new pasted object are inherited from the copied object. These layouts will automatically arrange the new object, vertically or horizontally, thus maintaining a homogeneous distribution.*

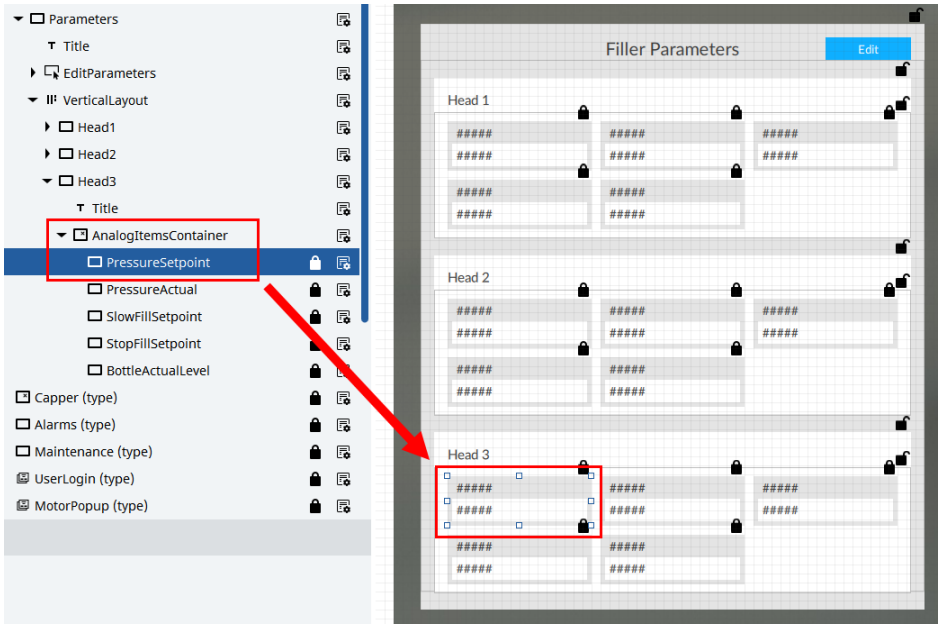
6. Thanks to the VerticalLayout, the new object ***Head3*** is auto resized to fit inside the ***Filler Parameters*** rectangle. In this case it will be renamed automatically (Head2 to Head3) because objects that contain a number at the end of their name are automatically renamed with the consecutive number.



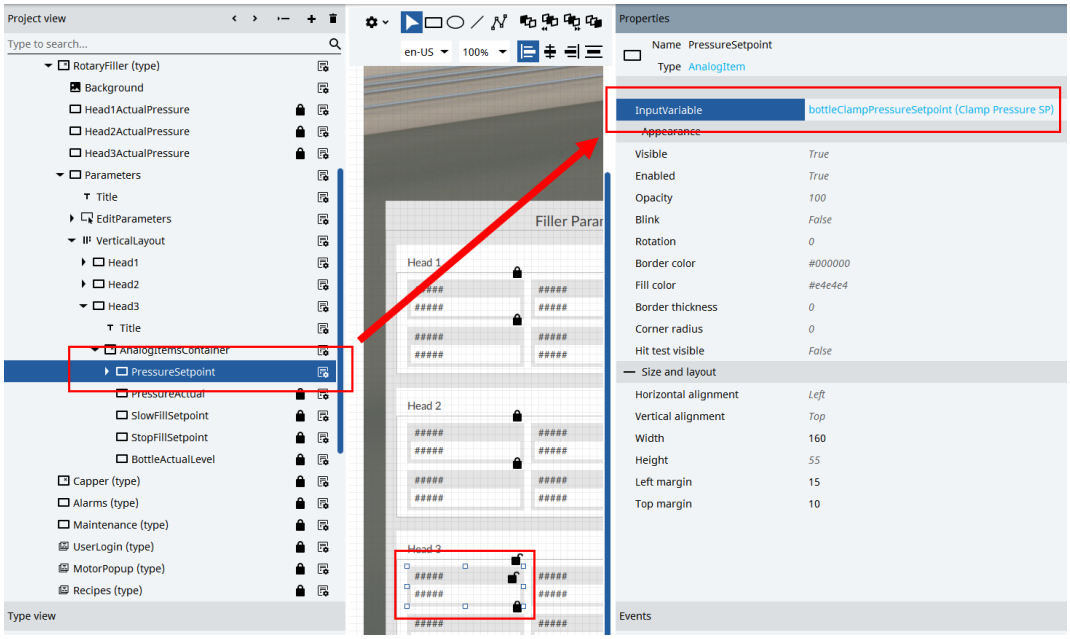
7. *Open* the node ***Head3*** and *select* the label ***Title.*** *Keeping* the selection on the label Title, change the property ***Text*** in the **Properties** panel: Head 2 to ***Head 3***.



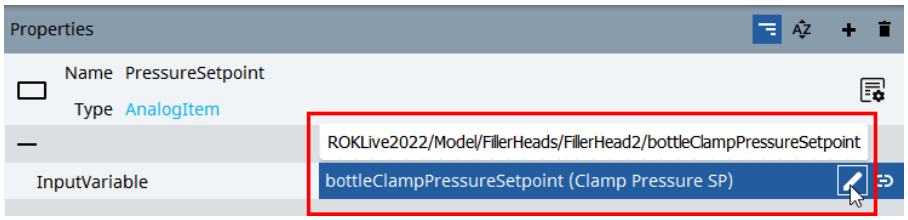
8. *Open* the node ***AnalogItemsContainer*** and *select* ***PressureSetpoint.*** This object corresponds to one of the five AnalogItems instances present in the AnalogItemsContainer.



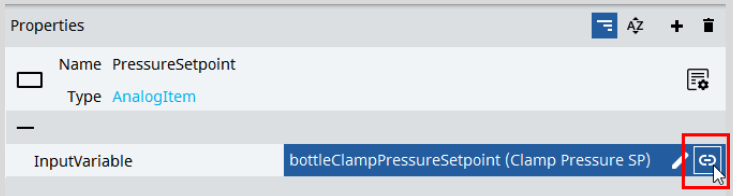
9. Now the variable bound to ***InputVariable*** must be changed: the old variable linked to Head2 is still linked to Head3.



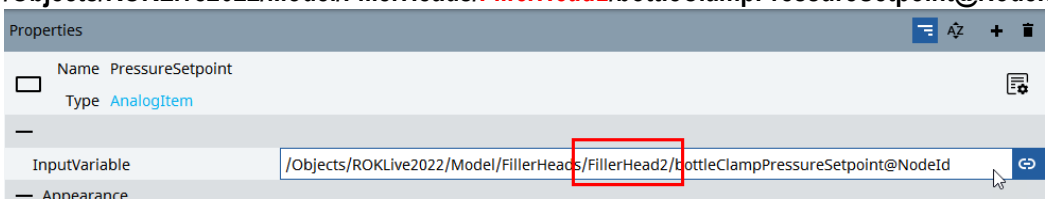
10. The link to the variable can be modified with a click on the ***pencil icon*** (modify)***.*** So, it is possible to change the link by modifying the path to the variable.



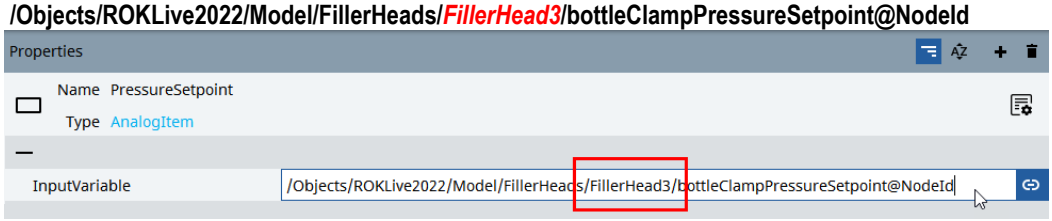
The link to the variable can be changed also opening the DynamicLink Popup with the ***chain icon***



11. *Modify* the path from this*:*
/Objects/ROKLive2022/Model/FillerHeads/*FillerHead2*/bottleClampPressureSetpoint@NodeId

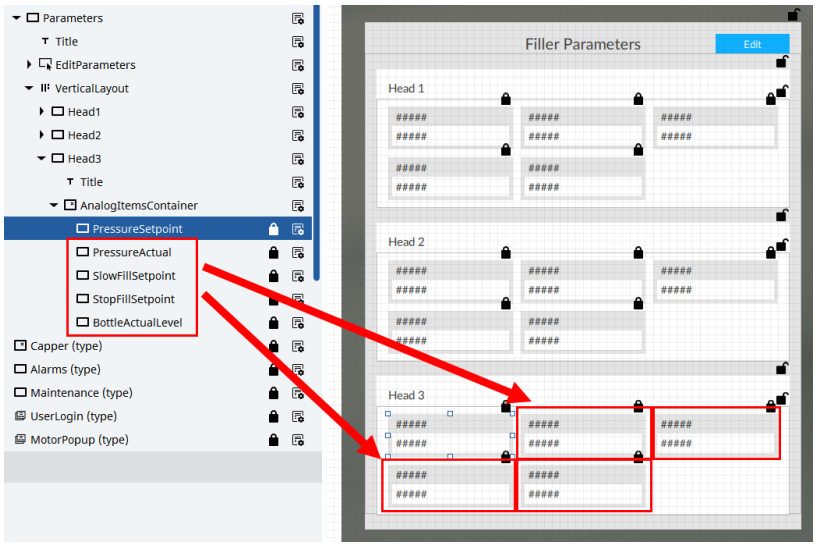


To this:
/Objects/ROKLive2022/Model/FillerHeads/*FillerHead3*/bottleClampPressureSetpoint@NodeId



When you changed the number from 2 to 3 make sure to press the **Enter** key.

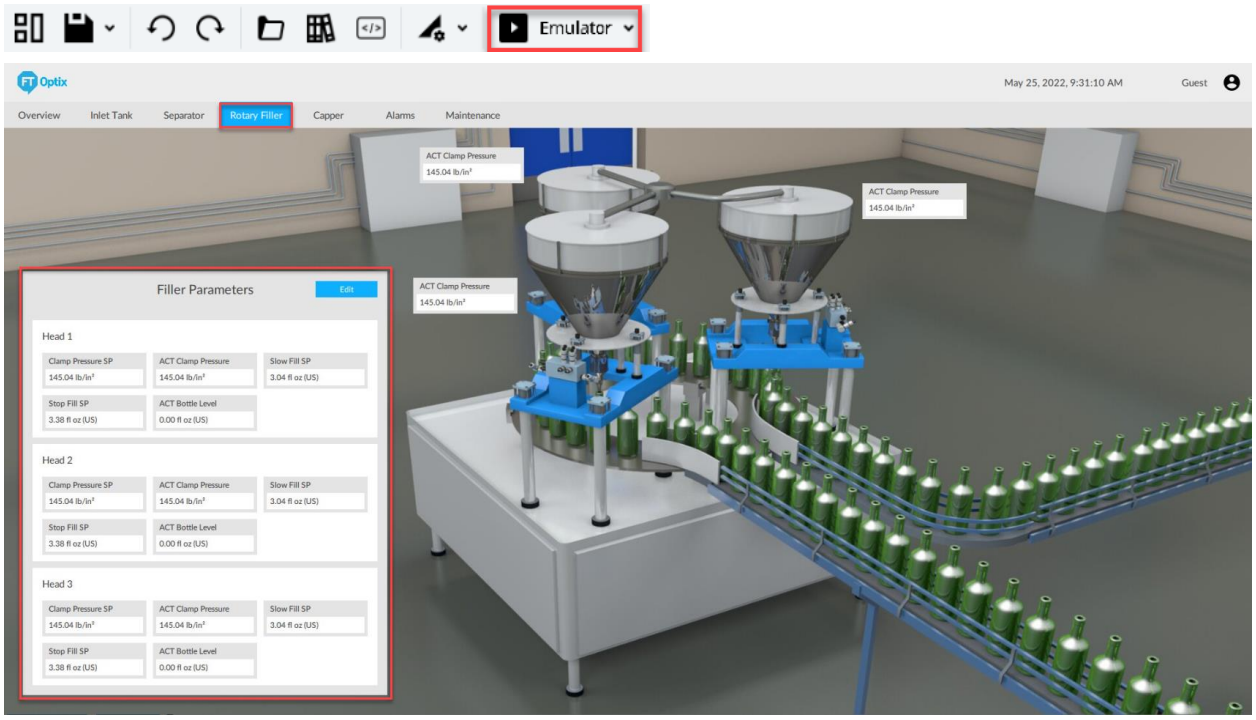
12. *Repeat* these steps also for the four remaining objects:
PressureActual
SlowFillSetpoint
StopFillSetpoint
BottleActualLevel.



> ***DynamicLink with pointer –*** *Consider that FT Optix allows defining a path with placeholders this makes the path dynamic. Especially useful when working with data structures that work with indexes.*
>
> ***E.g.*** */Objects/ROKLive2022/Model/FillerHeads/**{0}**/bottleClampPressureSetpoint@NodeId*
>
> *Where **{0}** is a variable that can be dynamically change.*

The configuration of the new Head3 object is completed.

13. Now run the **emulator** and click on the **Rotary Filler** tab.



Note: The values of the 3 heads might be similar, that is because they all use the same recipe, We'll change that later in the lab.Close the **emulator** when you have finished testing.

Style Sheets

Read about FactoryTalk Optix™ Style sheets

Introduction

*In FactoryTalk Optix™, a **Style sheet** object makes it possible to globally set some style properties of all graphical objects in the project, or of specific object classes (for example, the properties of **Switch** objects). The style properties are then inherited, starting from the style sheet, by the different types and instances in a project.*

Style sheets can be thought of as the same as Themes in windows.

Multiple style sheets can be created in the same project. This can be useful, for example, to allow the user to switch between a light tone interface and a dark tone interface at runtime.

Note

*FactoryTalk Optix™ includes some style sheet templates, available in the templates window (see **Use object and variable templates**).*

Inheritance of style properties

If the same property is set at more than one level, for example, if the text color is set in both the style sheet globally and on a custom Label object type, the property is given a value on the single object according to this order of decreasing priority:

property set on the instance

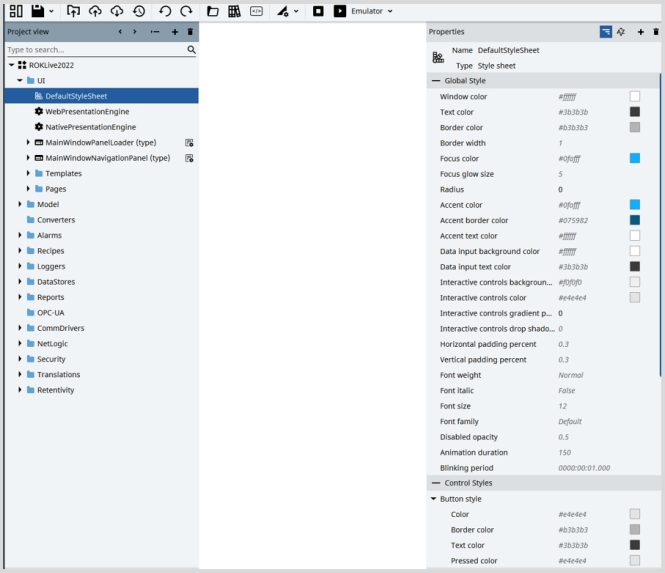
property set on the type from which the instance derives

property set in the style sheet for the specific object class

property set in the style sheet globally

If the value of a property is set to Default, the value corresponds to the value set at a higher level, in the same order of priority as above. For example, if the value of Text color is Default on a Label instance, the value set is the first one found for the same property at the type, object class or global level, in that order.

Example properties that can be set with a Style sheet.



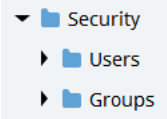
Security

In this section you will add a user and use a wizard to easily configure users and groups.

You will also see how easy it is to import widgets to configure users and groups online and how to keep these users even when you download an updated version of your application.

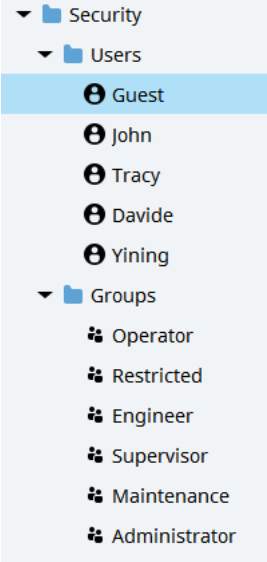
> ***Domain –*** *In this section of the lab you will use local users but FactoryTalk Optix also has the ability to use users from a domain controller.*

1. Open folder/Node **Security**

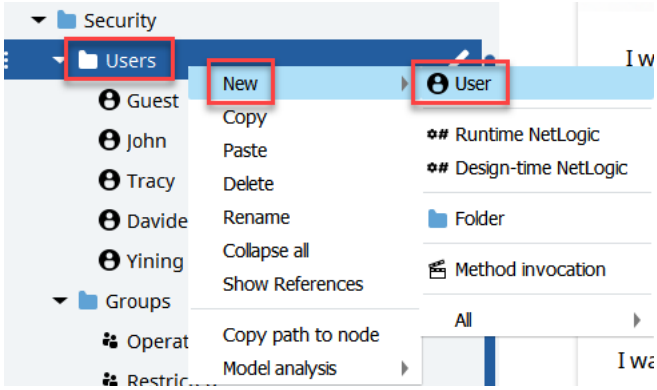


2. Open **Users** folder/node and also **Groups** folder/node

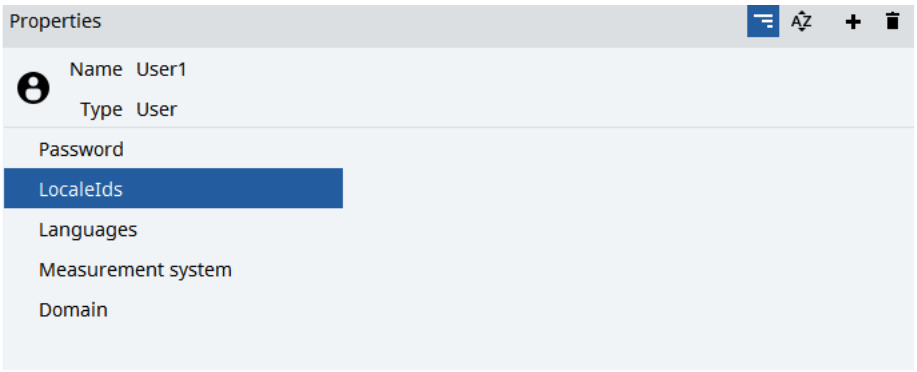
   As you can see, some users and groups for this project have already been created



3. *Right click* on the **Users** Folder and the select **New** > **User**



4. On the **right side** you will see that the **User1** Properties are shown.



Read about Locale

Introduction

The term locale means the set of display settings of a user interface based on language and country. It is represented by a label called locale ID, made up of language and country (e.g., en-US, en-UK, it-IT etc.).

A project can support multiple locales. The locales supported by the project are set in the ***Locales*** *property of the project node.*

In multilingual projects, the Presentation engine displays the interface based on the session locale, if configured, and the texts displayed based on the translations available in the ***LocalizationDictionary*** *object.*

Locale IDs

*The Locale ID specifies a language and a country, e.g. en-US, en-UK, it-IT. In particular, the second segment determines the date and time format, the date separator and the measurement system (****International Measurement System****,* ***United States Customary System*** *or* ***British Imperial System****).*

Session locale

The session locale determines the locale of the user interface (i.e. the translation of the texts according to the project settings), the data display format, and the conversion of all values according to the required measurement system. It is set at runtime based on the user or object locale ***UI Session Session UI****.*

Note

If there are no locale settings at the user or Session UI object level, the session locale is set based on the locales supported by the project: in particular, the first locale in order of writing in the ***Locales*** *property has priority (see* ***Fallback locale****).*

Fallback locale

If the user-determined session locale is not configured in a project, or if translations of some interface texts are not available for the session locale, the texts are displayed in the project fallback locales, configured in the ***Fallback locales for translations*** *property of the project node.*

Note

When there are multiple fallback locales, at runtime the system uses the fallback locale based on the order of insertion of the different locales in the ***Fallback locales for translations*** *property. If the list of fallback locales contains the locale not supported by the project, this fallback locale takes priority over the others.*

User locale

The user locale is personal data used at runtime to set the session locale. It is set in the ***LocaleIds*** *property of the objects* ***User User****.*

Through C#, using the APIs that return nodes (see ***Access to project nodes****), it is possible to set multiple locales for the same user, i.e. alternative locales if the FactoryTalk Optix™ Application does not support the main locale. The order in which the user locales are shown determines their priority for setting the session locale.*

When the session locale is set automatically at runtime based on the user locale, if none of the user locales are configured in the FactoryTalk Optix™ Application, the session locale is set based on the project fallback locales.

Locale through Session UI

The ***UI Session Session UI*** *object is useful for managing the session locale in multilingual projects for which there is no information on the user locale, and therefore the user is asked to choose the locale at runtime. For the configuration, see* ***Configure the selection of the locale at runtime through the Session UI object****.*

Customized combinations of language and measurement systems

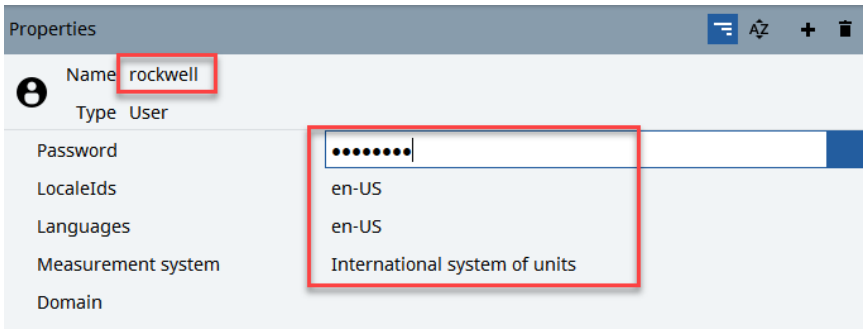
The traditional locales determine the measurement system based on the country indicated in the locale ID. In any case, through the ***Session UI*** *and* ***User*** *objects, it is possible to associate a measurement system of your choice with a language supported by the project. For example, it is possible to select Italian (it-IT) as the display language, but the US system of measurement (****United States Customary System****), which also determines the data display format.*

Customized measurement systems

At runtime, FactoryTalk Optix™ displays the data relating to the physical quantities in the measurement system required by the session locale.

In any case, it is possible to display the physical quantities in units of measurement other than the standard units of the reference metric system, or create customized units of measurement.

5. Change the ***User1*** to ***rockwell*** with password ***rockwell***, LocalIDs and Languages to ***en-US*** and Measurement system to ***International system of units***.



Tips :

The Domain is empty since a domain controller is not used in this lab

For the Measurement system, you have 3 options

International system of units

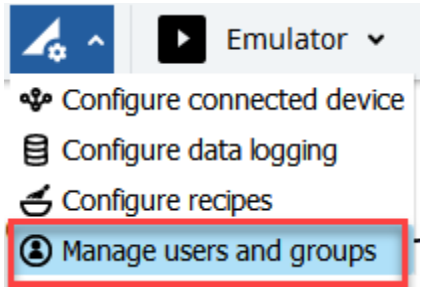
US Customary measurement system

British Imperial units

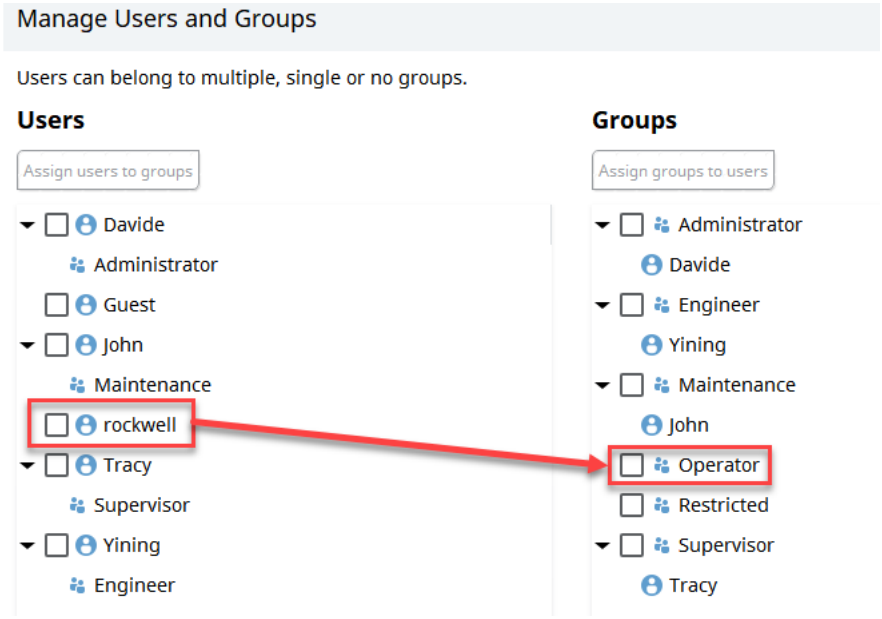
This measurement system is used to change the engineering units automatically when a user logs is.

You will see this this when the application is run.

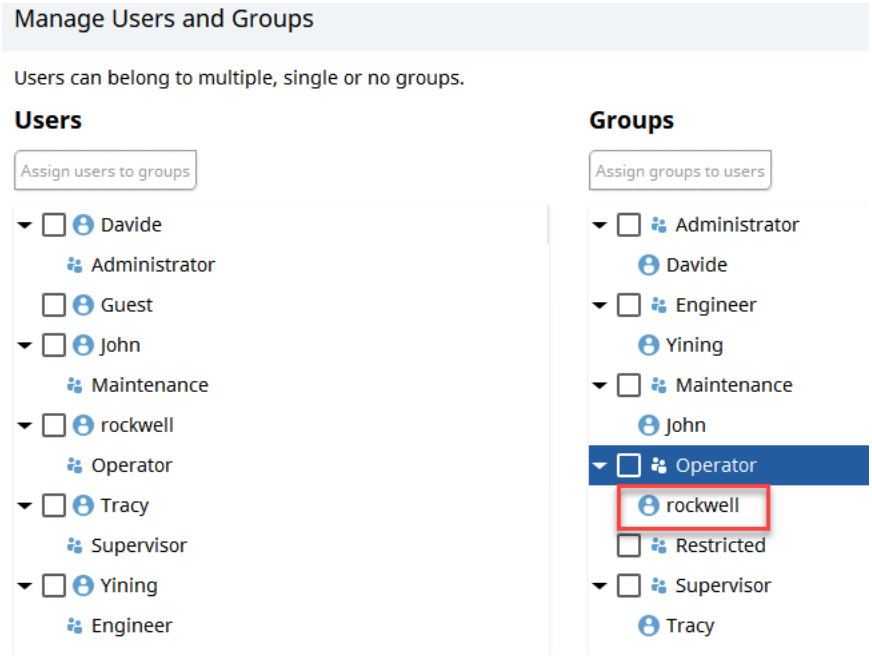
6. Now you will use a wizard to assign the users to the security groups. Click on  next to **Emulator**.
7. Select ***Manage users and groups***



8. Next you will assign the user ***rockwell*** to the operator group. *Select* the user ***rockwell*** and drag it to the ***Operator*** group.



9. You can see that the user ***rockwell*** is now user the ***Operator*** group.



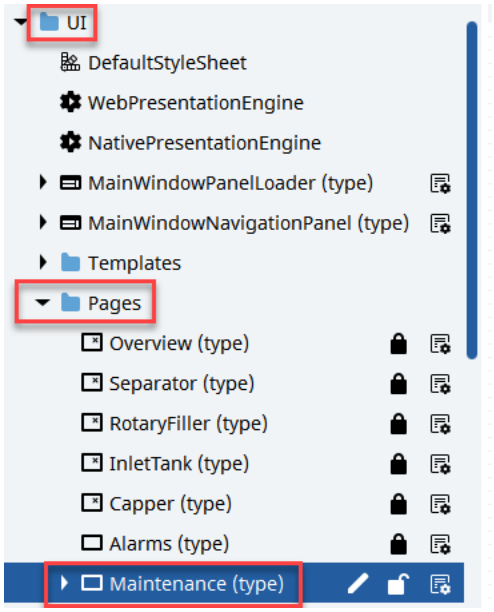
There are 2 other ways that can be used.

When you select a user and click on the button ***Assign users to groups***, you can select which groups you want the user to belong to.

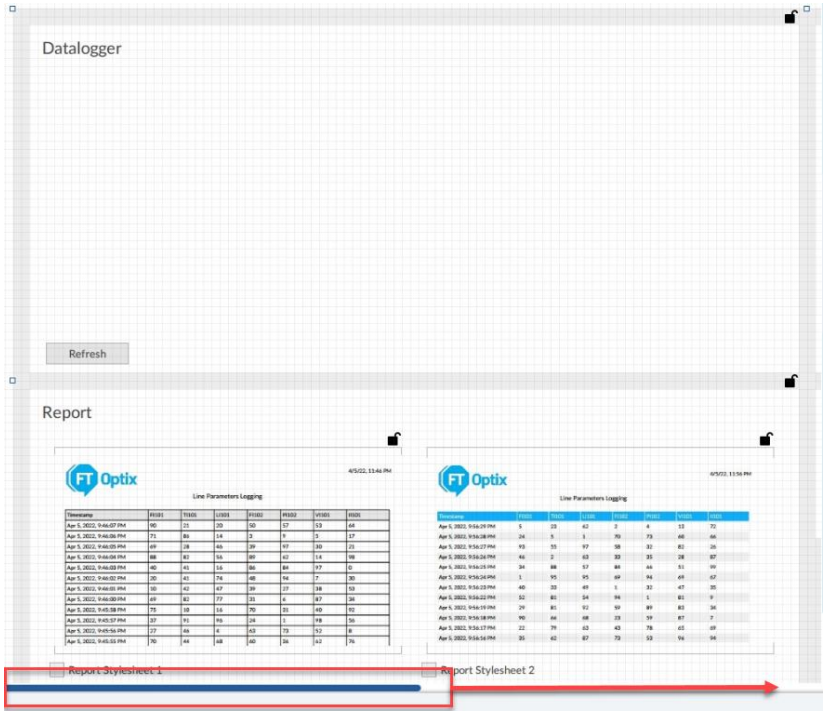
When you select a group and you click on ***Assign groups to users***, you can select the users that you want to move into the selected group.

You can also select multiple users and groups and use the assign buttons.

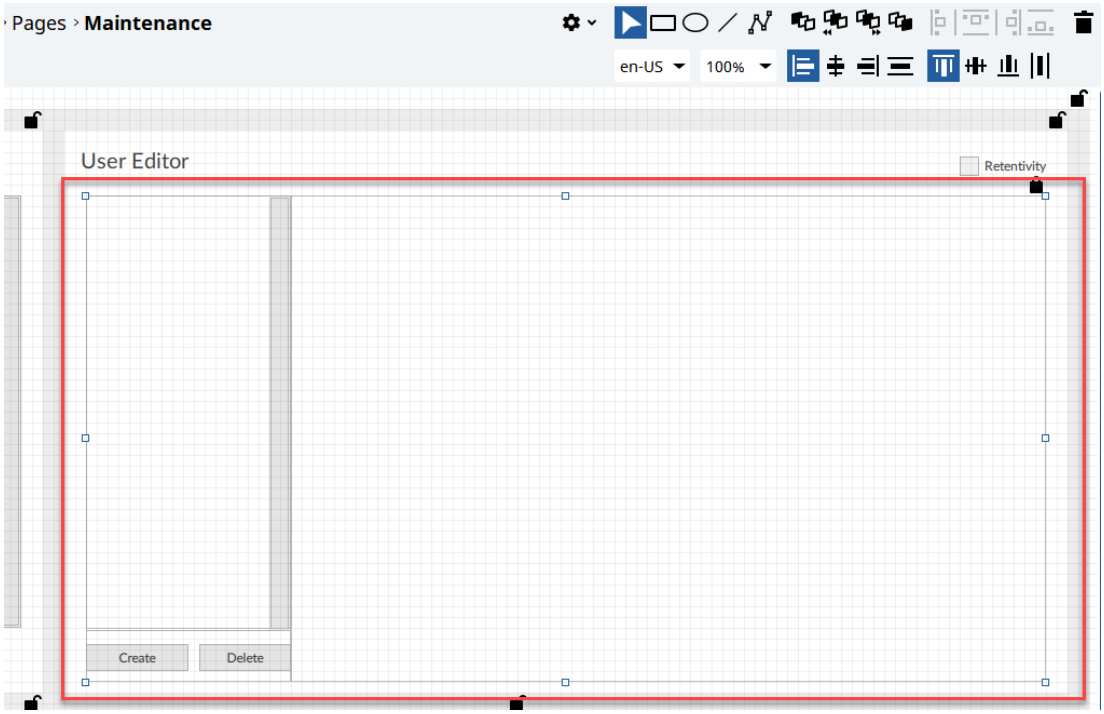
10. Go to ***UI*** > ***Pages*** and *double click* on ***Maintenance*** (type).



11. On the screen use the ***bottom scroll bar*** to move to the *right*.



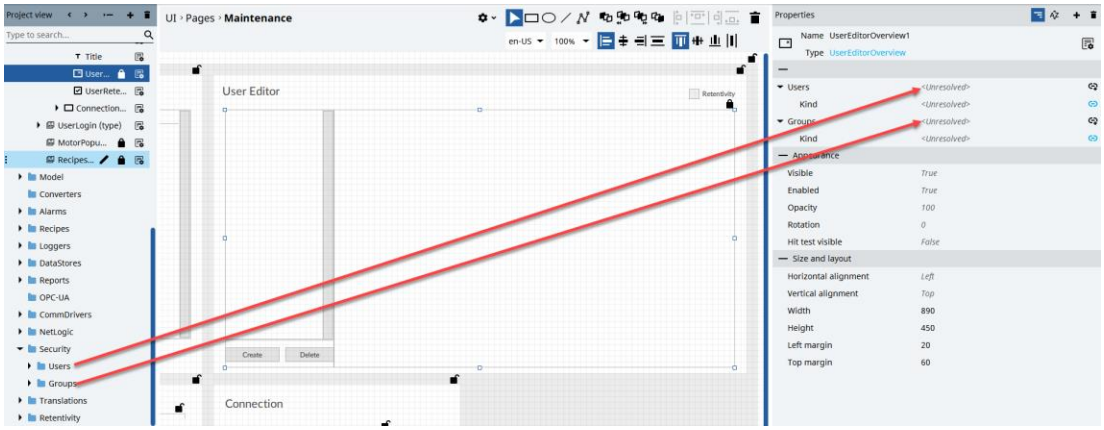
12. Click on the object as shown below in the ***User Editor***.



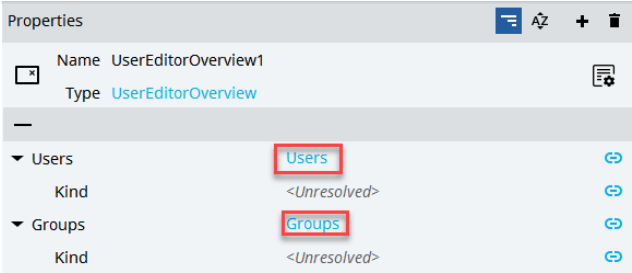
13. Now you will move the ***users*** and ***groups*** into the properties of the object you just selected.

    In the left column, go to **Security > Users**, select this folder and drag it to the **Users** in the properties of the object.

    Do the same for the Groups to the Groups Property.



14. The properties of the object should look like this.

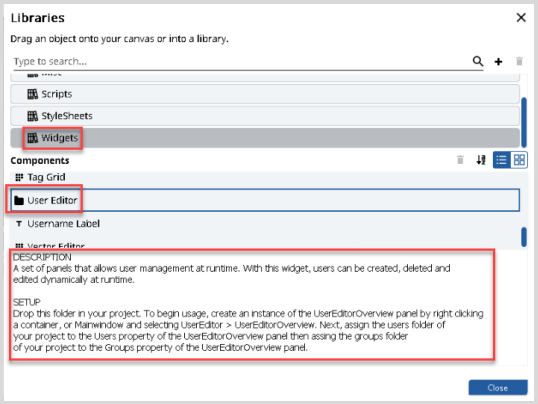


What you have done now is configure a standard object from the template library called Widgets and linked Users and Groups to all the objects within this widget.

When you click on the Template Libraries icon  *then you will see a popup open with some template scripts, stylesheets, GraphicElements, Misc and Widgets.*

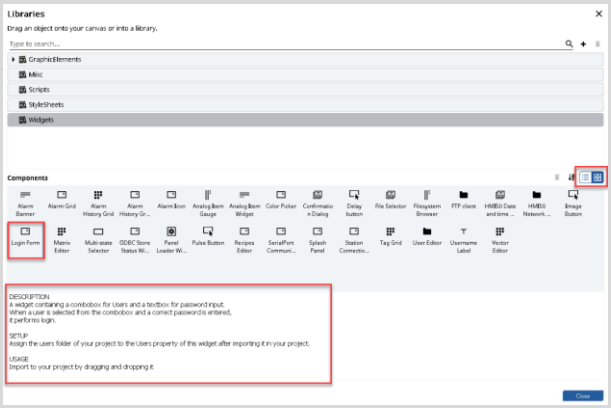
*The **search** bar makes it easy to find anything in the library.*

When you click on Widgets and then user Editor, you can see what needed to be done to implement this widget into your application.



The login form that you will use in your application is also available in these widgets.

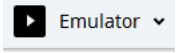
Using the List or Tile switch, you can switch between a list or icons for all the objects.



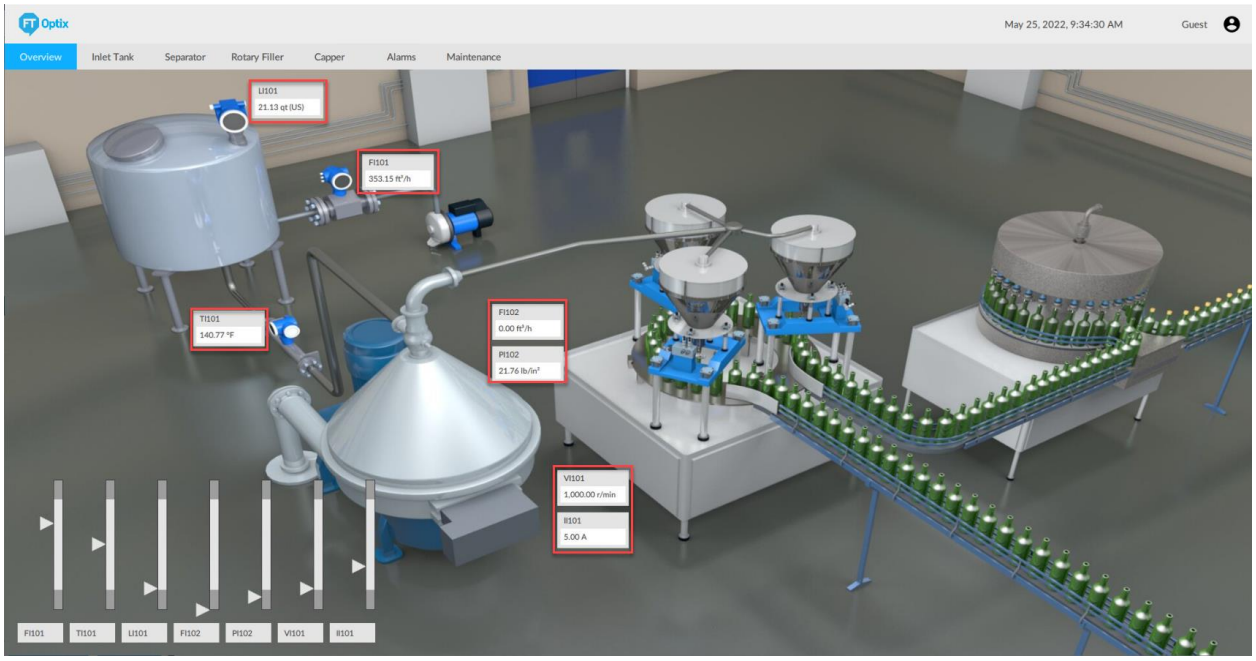
Since all the objects are already included in the application you will not add them so you can just close the Libraries if you opened it.

Now you will test the modifications you have implemented so far.

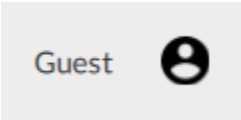
15. In the top bar, Click on Emulator.



16. The application will start, look at all the grey boxes you see on the screen. Specifically at the values and the engineering units.



17. *Click* on the Login icon.

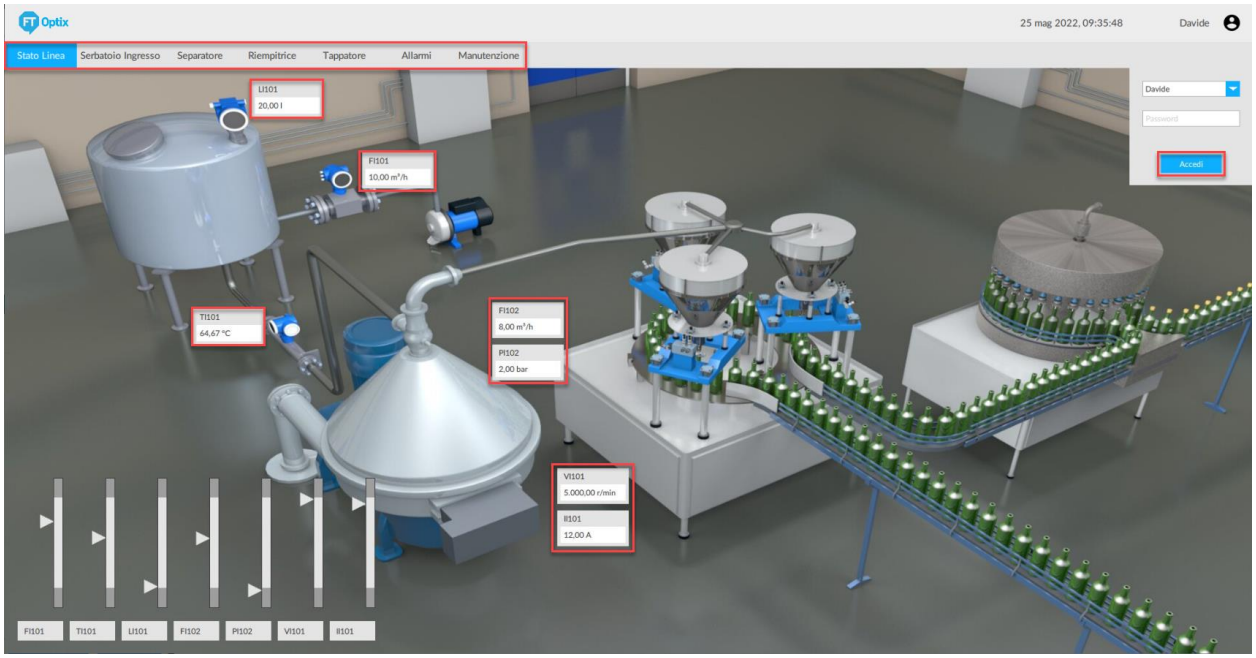


.

18. A drop-down menu will appear, *click* on ***Guest*** user and change to ***Davide*** and *click* on ***Login.***

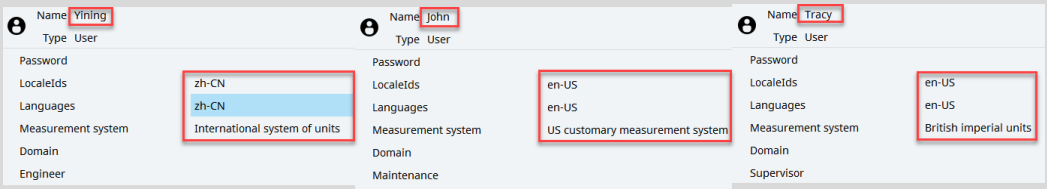
    With the login 2 things changed.

    1. All the text fields and menu are changed to Italian.
    2. All the values and engineering units changed to that which are commonly used in Italy as measurement system.



For each user, you define language and measurement system.

<u>Measurement system</u>

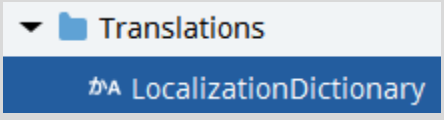


You can select between 3 different measurement systems : US customary measurement system, British imperial units and International system of units.

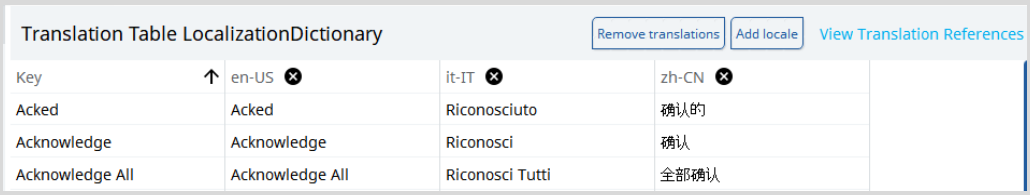
You can always modify certain measurements or add a new measurement system if that would be needed.

<u>Languages</u>

Under the folder Translations you can see the Localization Dictionary.

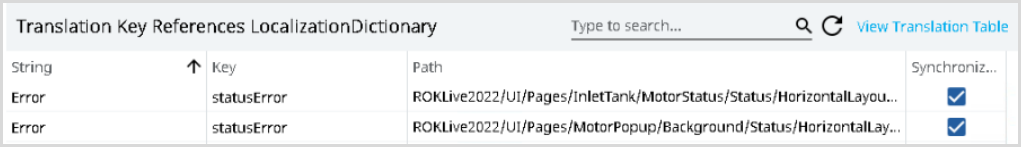


In here you can add Locale or languages and also do the translations.

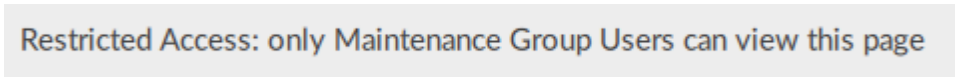


Every text field in the application gets a unique name called Key which is the first column you see in the table.

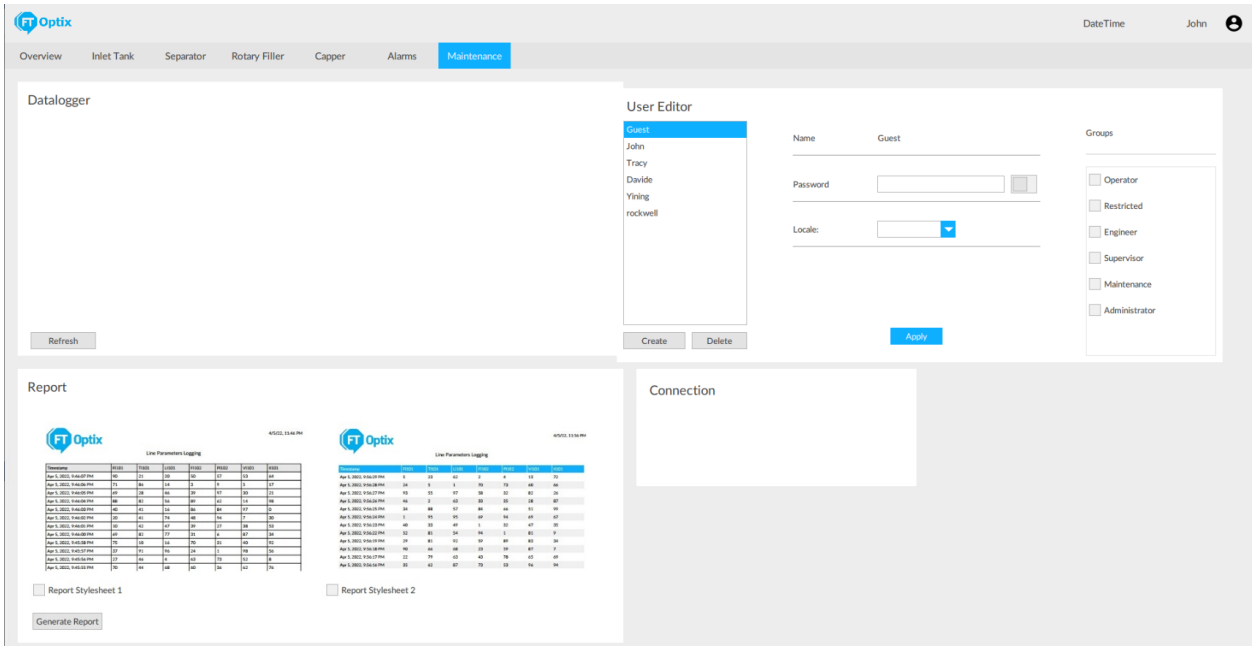
When you click on View Translation References, you will see all they strings used in the application and each string has a unique Key. So even if for example Start is used multiple times in your application, it will only be shown as 1 key so you only need to translate it once.



19. You can switch to other users to see what happens; all users are created without password only the user ***rockwell*** has password ***rockwell.***
20. *Login* as user ***guest*** and *click* on the ***Maintenance*** tab in the menu. You will see a message on the screen that you don't have access to this page. The page was configured that only John has access to the Maintenance display.

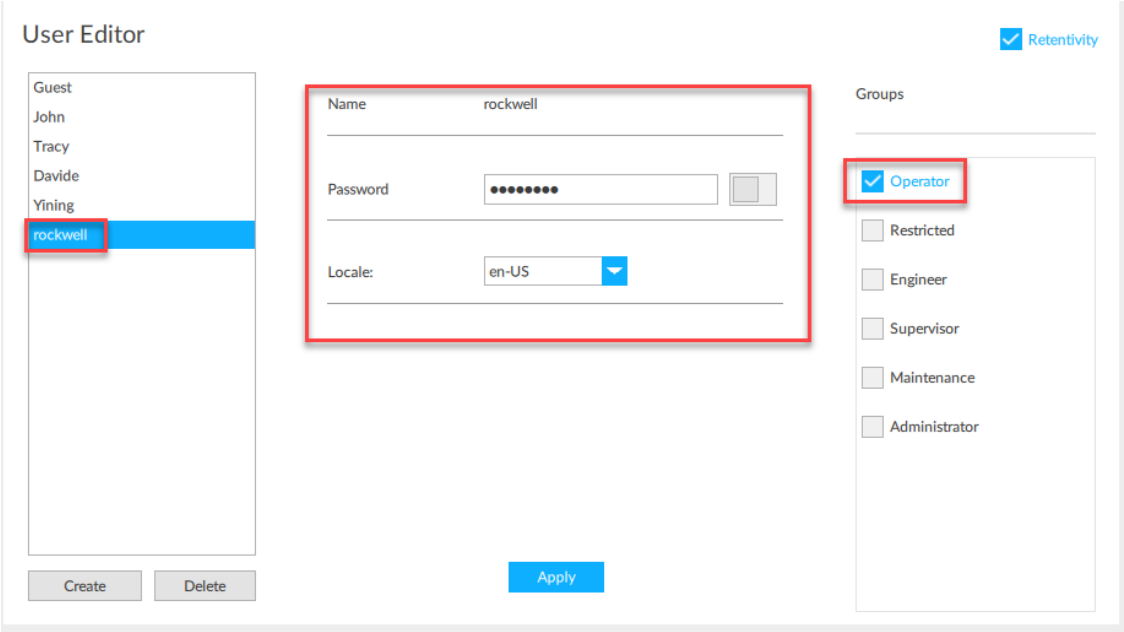


21. *Login* as ***John*** who is part of the maintenance group has access to Maintenance.

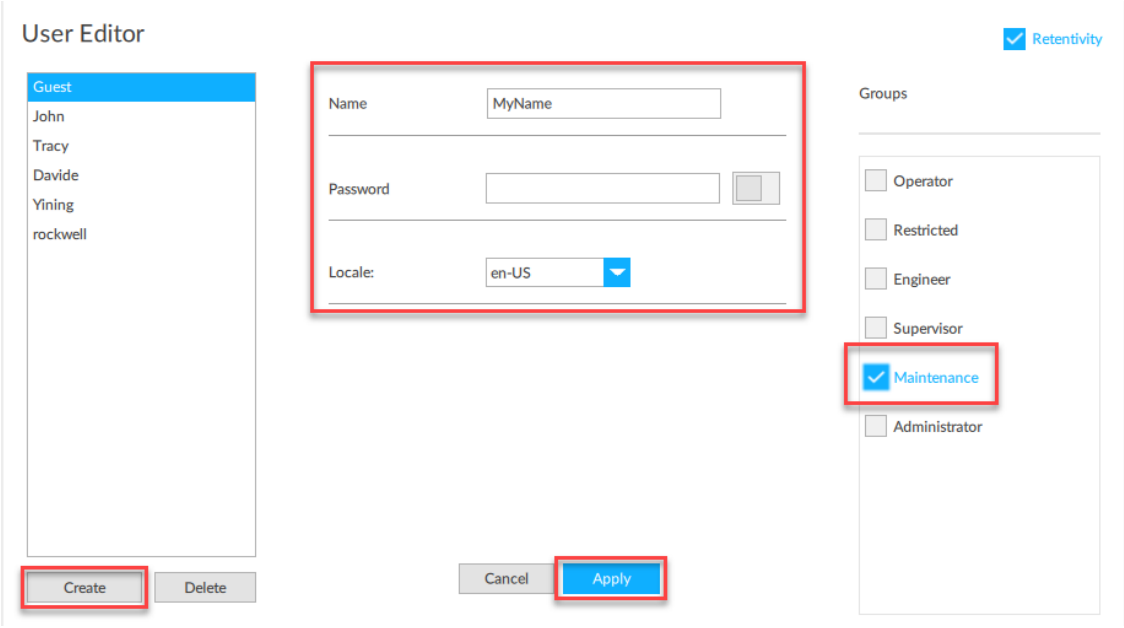


As you can see on the, all the users in your application are included on this User Editor object.

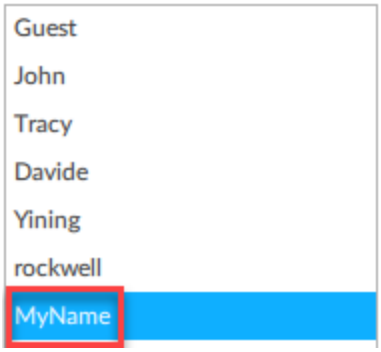
22. *Click* on ***rockwell*** and you will see the password is filled in and that he is part of the **Operator** group.



23. *Click* on ***Create*** and create a new user using your own name and assign yourself to **Maintenance** group so that you have access to this Maintenance screen. When finished, click **Apply**. Make sure to press on the **Enter** key when you have filled in your username and password.

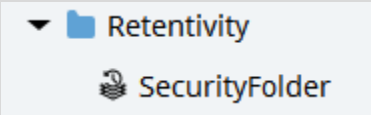


24. The user that you created is now in the application.



When you close the application and run the emulate again, all the users you online added will still stay in the application.

The reason that this is possible is because a Retentivity database has been added to the application.



In this case it is used it for security but this retentivity database can also be used for internal tags.

Recipes

In this section of the Lab you will configure recipes using a widget.

Read about Recipes

Introduction

In *FactoryTalk Optix™*, recipe management is done through one or more Recipe schema objects.

A Recipe Schema object defines a set of variables, or ingredients, with which it is possible to configure different recipes, i.e. different sets of values for the same set of variables.

This object can be used to define the values of a configuration and save them to be reapplied as needed. It can be useful, for example, to restore the initial settings of the machine following changes that compromise its operation.

Ingredients of a Recipe Schema

To define the ingredients in a Recipe Schema, a destination node must be set at design time that contains, among all the child nodes, the variables of interest. It can be any node of the project and the choice depends on the structure of the project and its complexity. For example, the destination node can be the Model folder, an object inside the Model folder, or a node that contains variables of one or more PLCs.

Hint

To define variables contained in different nodes of a complex project as ingredients of a Recipe Schema, create a dedicated object in the Model folder and add variables inside it, then create a dynamic link between the variables and desired ingredients. In fact, if the object is selected as destination node, the selection of the Recipe Schema ingredients is more intuitive, as only the variables referenced in the object are shown.

Recipe management at runtime

To design recipe management at runtime, the Recipes Editor widget included in FactoryTalk Optix™ Studio can be used.

The widget can be used as provided or some of its components can be reused to design a customized solution.

Edit model

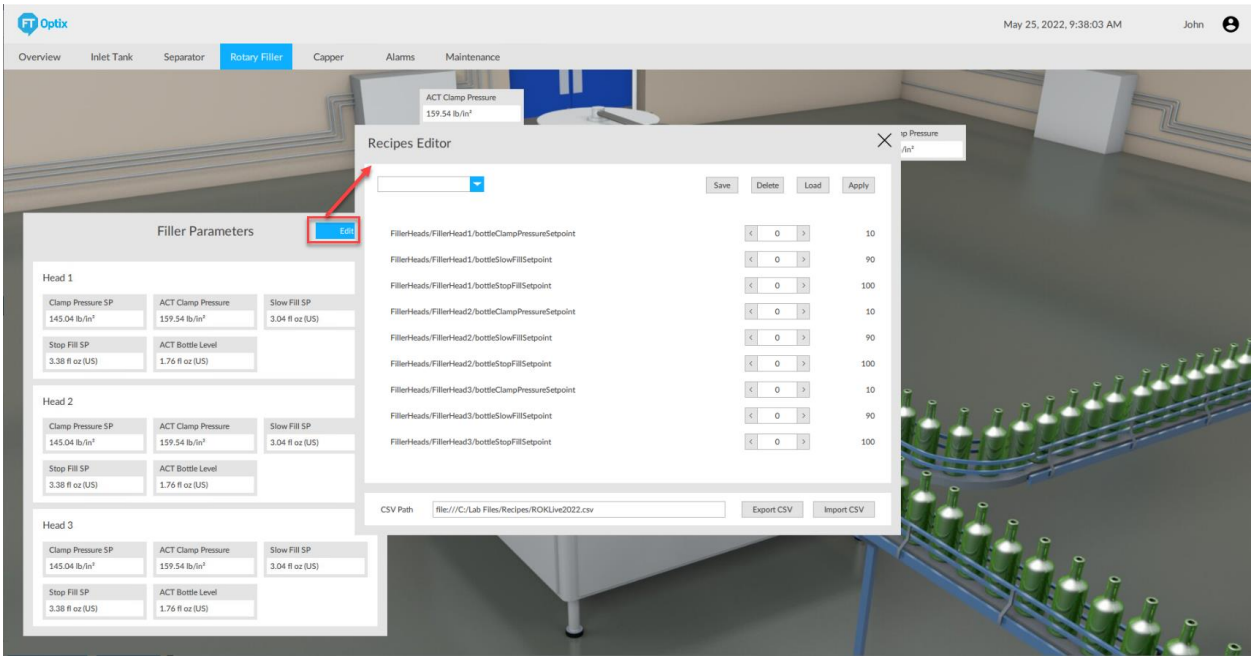
When the user creates, edits or loads a recipe from the database at runtime, an Edit model node is automatically created in the project root node that can be referred to at design time. The Edit model node contains a temporary copy of the created/edited/loaded recipe data, until it is saved, deleted or sent to the PLC.

Important

if execution of the FactoryTalk Optix™ Application is stopped, the Edit model node and the data it contains are deleted.

To design custom recipe management, the mechanisms related to the Edit model node must be known. On the other hand in the Recipes Editor widget, the references to the Edit model node are already correctly set to guarantee all features at runtime.

This will be the result:



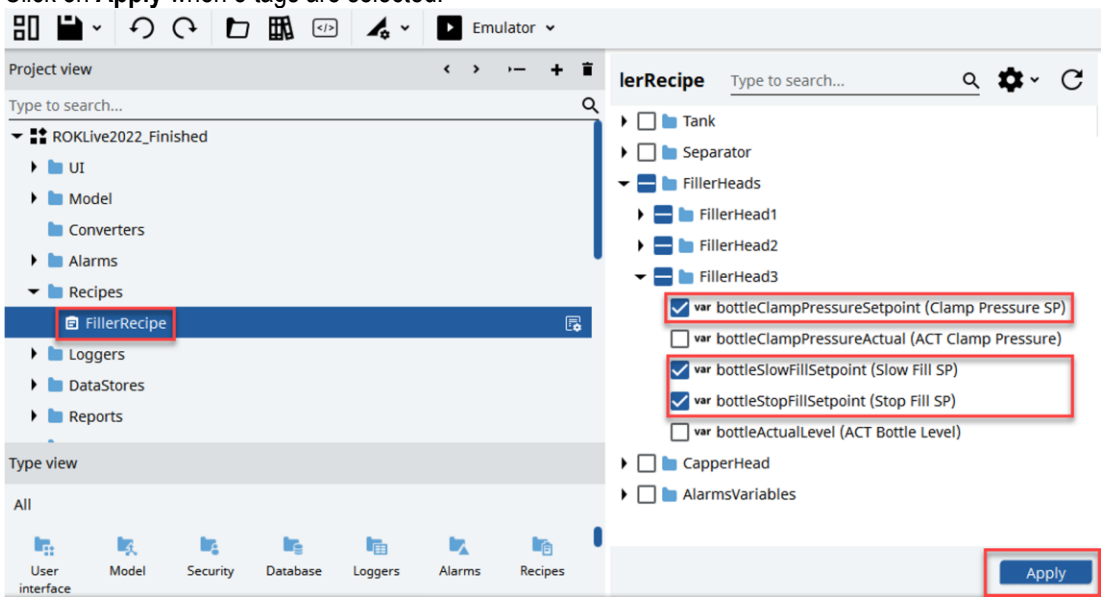
1. Adding Rotary Filler Head 3 parameters.

   Recipes parameters are stored in a database table that is connected to a RecipeSchema, this data block will manage a set of recipes containing all desired values.

   Expand **Recipes** and *double click* **FillerRecipe** to open the **RecipeSchema** editor, *expand* the variables tree and select the following under **FillerHead3**:

   - **bottleClampPressureSetpoint**
   - **bottleSlowFillSetpoint**
   - **bottleStopFillSetpoint**

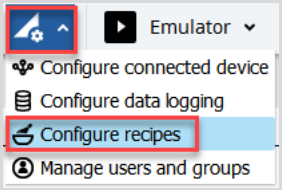
   Click on **Apply** when 3 tags are selected.



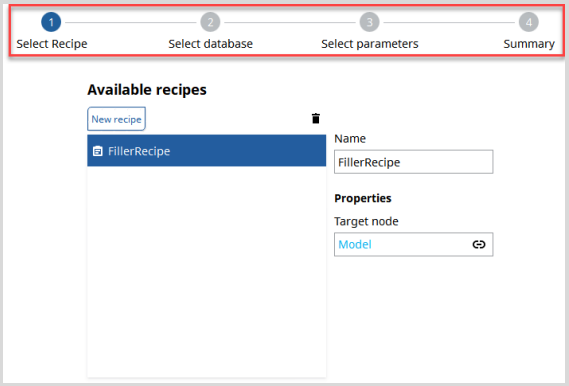
RecipeSchema parameters contain the used database where recipes tables are created, you can manually specify a Table name or leave that automatic (blank)

There is also a Recipe wizard available to create a new recipe.

This includes, database selection or creation and all the tags needed in your recipe.



Only 4 steps are needed to configure a recipe.



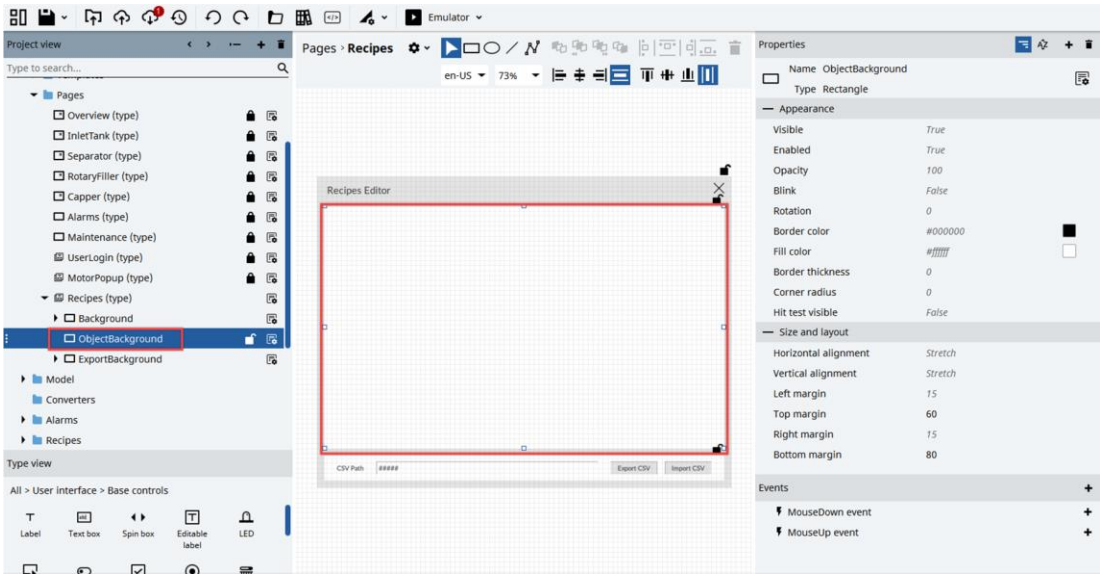
Configuring the **Recipes editor** widget

2. Adding Recipes editor to Recipe popup

   Browse to **UI > Pages** and *double click* on **Recipes (type)** to open the editor.

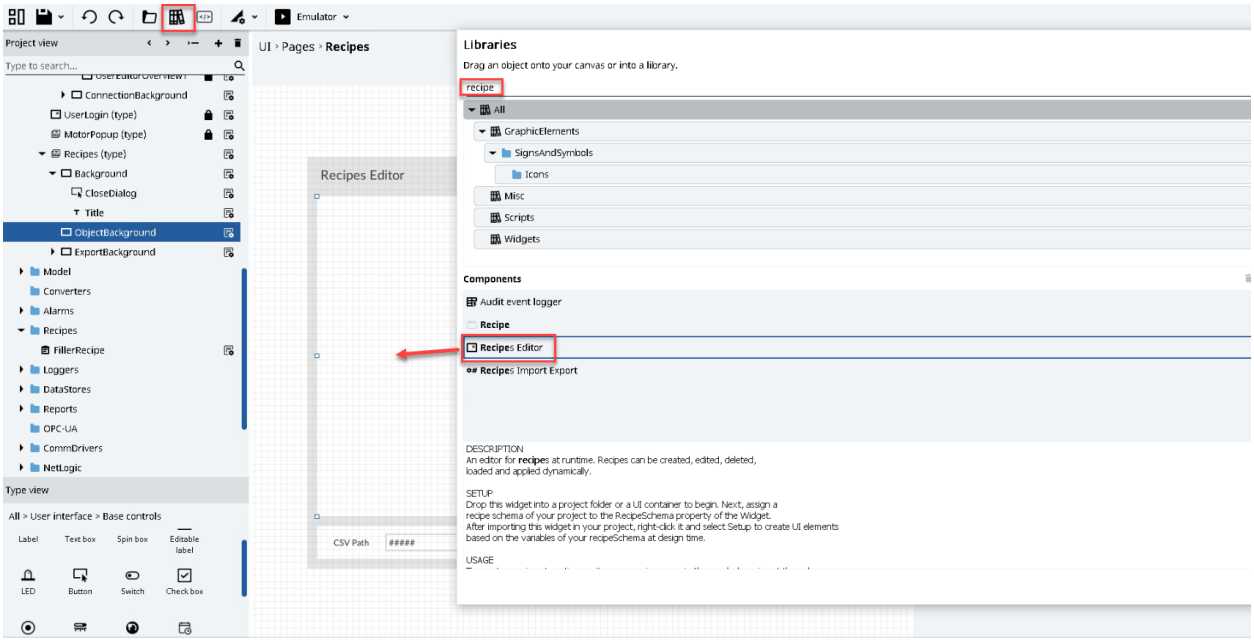
   The **Recipes** popup is divided in three spaces:

   - Header (title)
   - Main content (recipes editor you are going to add)
   - Footer (recipes import and export functionalities)



3. Importing the recipes editor.

   Open the **Libraries** and type in **Recipe** in the *search* bar and click enter, *drag and drop* the **Recipes Editor** widget to the blank box in the center of the popup. Close the **Libraries**. The widget has just been imported into the project.

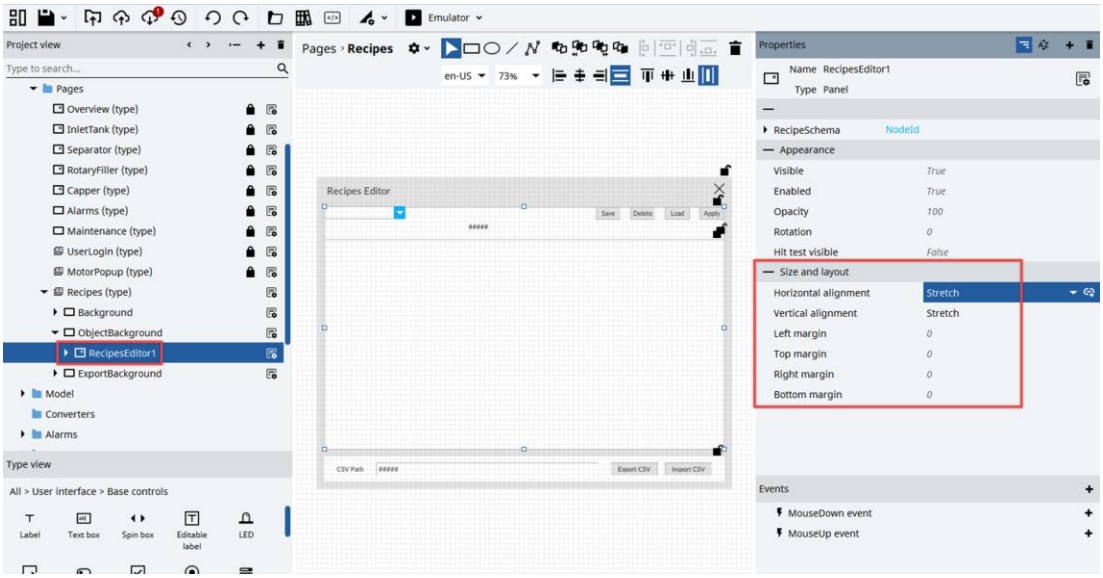


4. Adjusting the *RecipesEditor*.

   Click on **RecipesEditor1** object and set it **Properties** to:

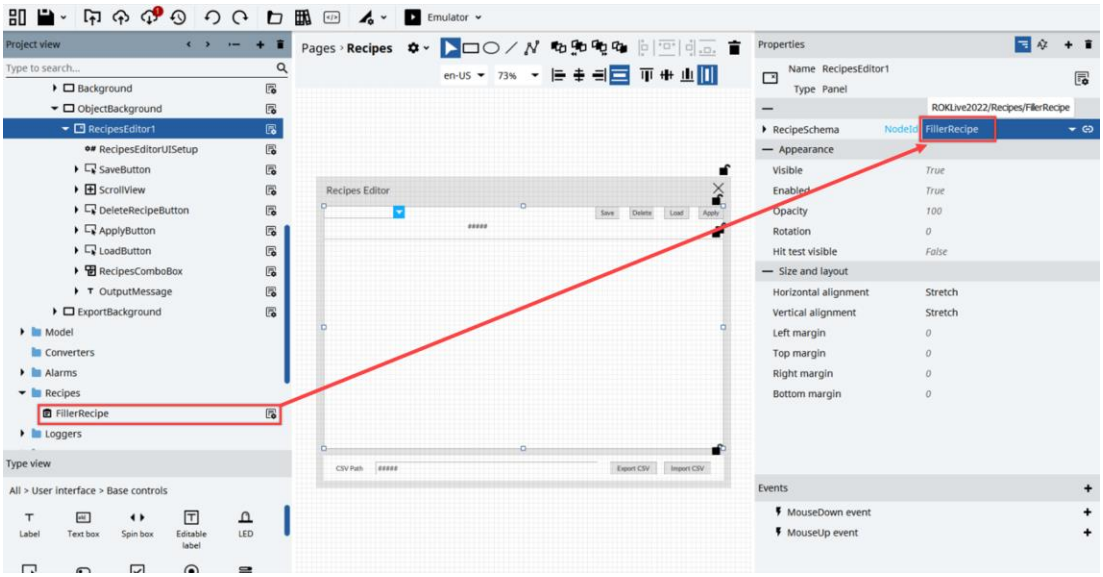
   - **Horizontal alignment:** ***stretch***
   - **Vertical alignment:** ***stretch***
   - **Left margin:** ***0***
   - **Right margin:** ***0***
   - **Top margin:** ***0***
   - **Bottom margin:** ***0***

> *Items in a project are arranged in containers and content, content parameters are always related to its container, if you set a content to stretch it will fit the whole area of its parent element*



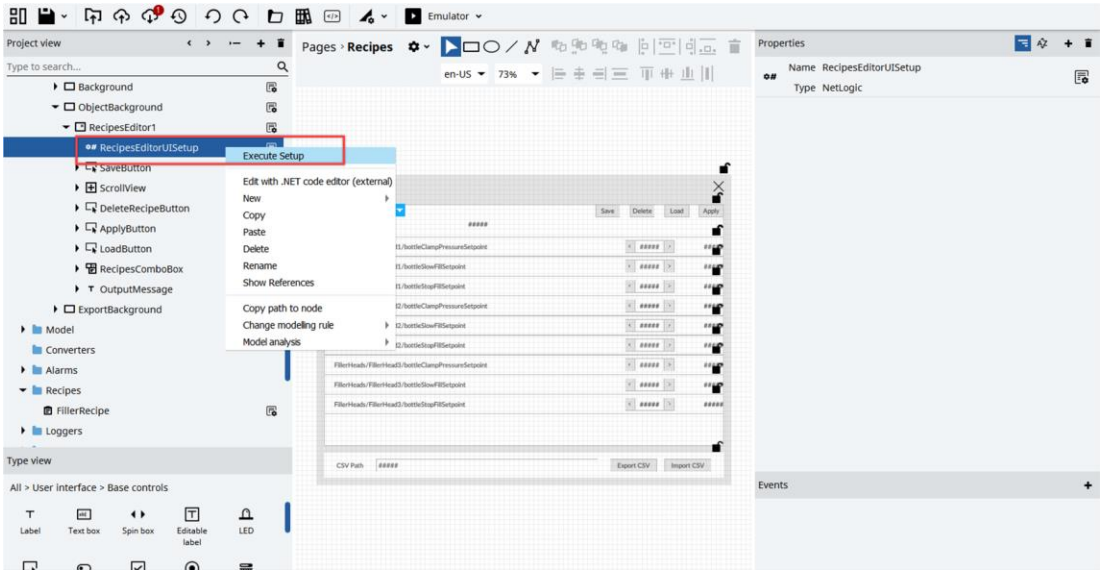
5. Attaching the RecipesEditor to a RecipeSchema.

   Drag and drop the **RecipeSchema** you previously configured to the **RecipeSchema** parameter of **RecipesEditor**.

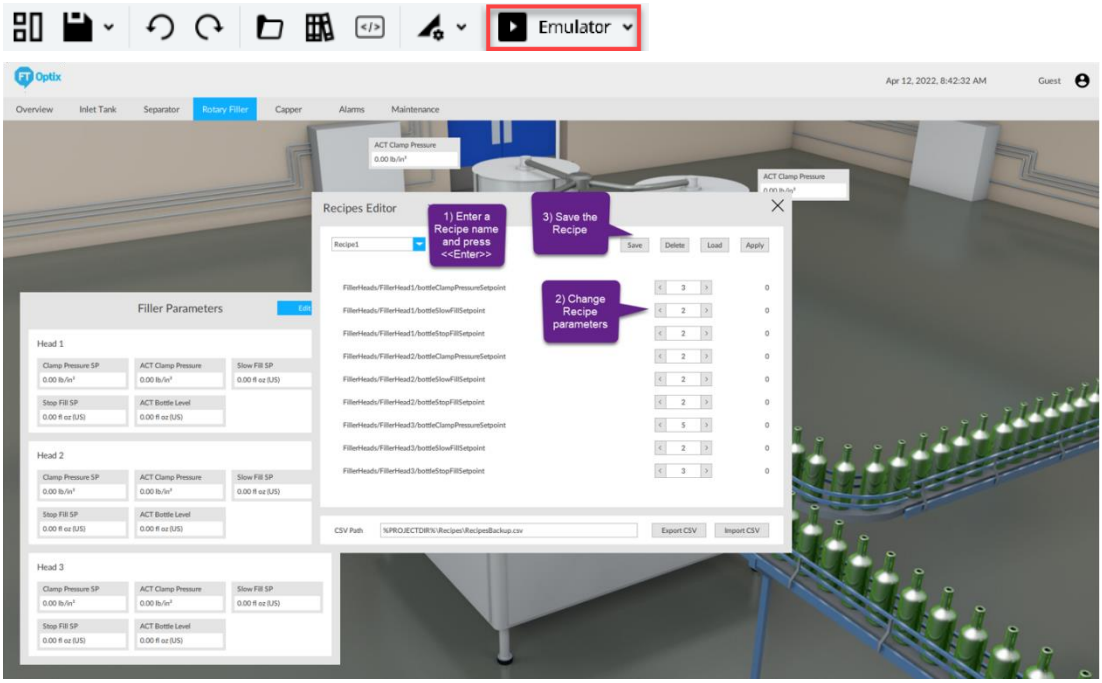


6. Populating the **RecipesEditor** widget.

   Expand the **RecipesEditor** you just configured and *right click* on the **RecipesEditorUISetup**, Select **Execute Setup** script to trigger the **Execute Setup** function; this will populate your widget with the whole set of variables you configured in the **RecipeSchema**.



7. Now run the **emulator**, *click* on the **Rotary Filler** tab and then *click* the **Edit** button.



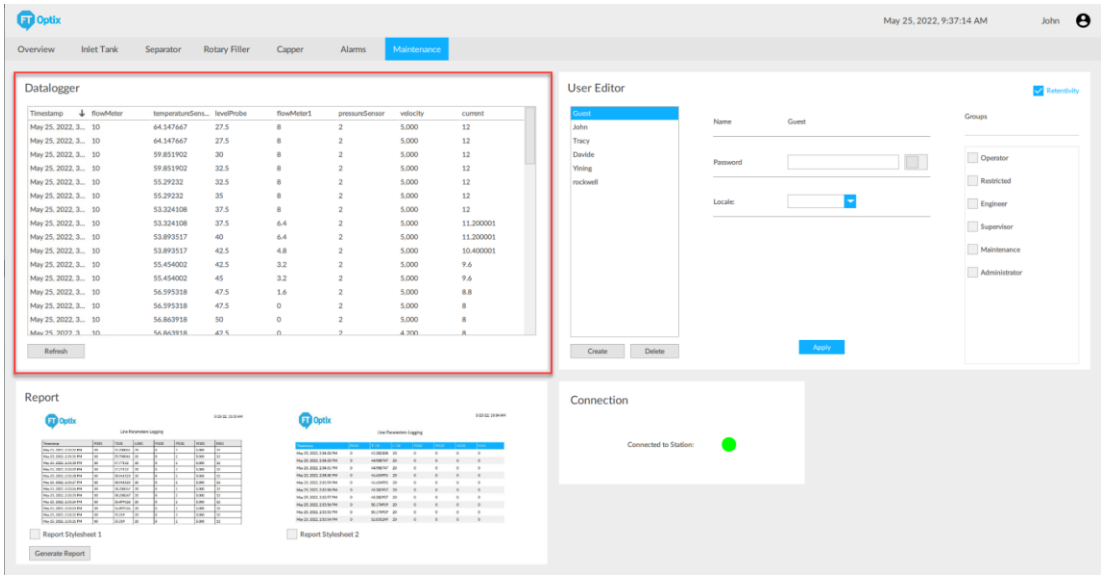
Button	Function
Save	Store the recipe in the database during editing.
Delete	Delete the selected recipe from the database.
Load	Load the recipe values from the PLC to the HMI application.
Apply	Apply the selected recipe on the PLC.

8. Close the emulator when you have finished testing.

Data Grid and Datalogging

In this section of the Lab you will add a DataGrid looking at an embedded database that is been populated with a Datalogger.

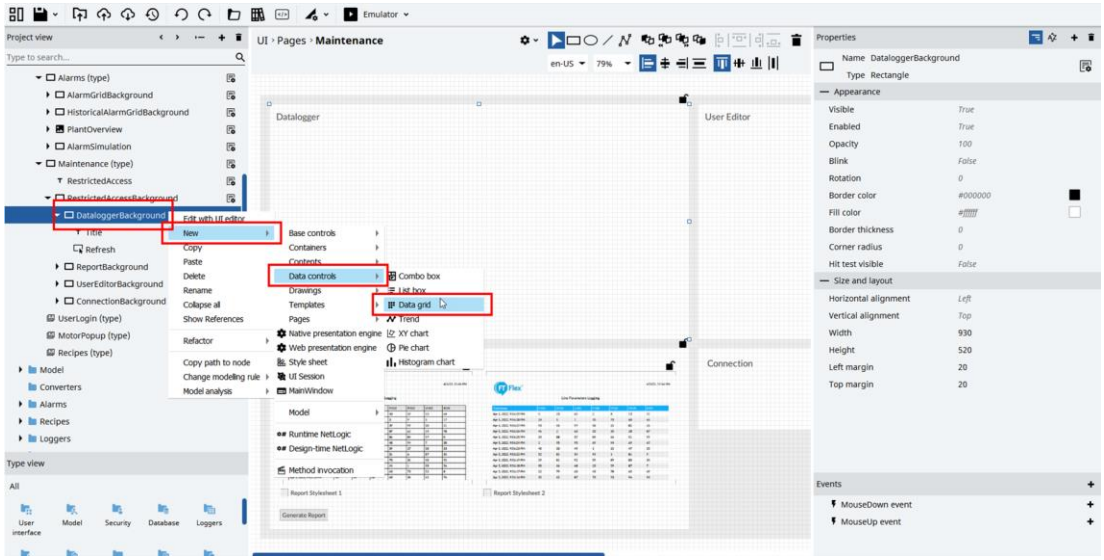
This will be the result:



1. Adding a DataGrid to the project

   Open the **Maintenance** page, this page is split into two pieces: a static background with a warning about being logged in to view the page and a “RestrictedAccessBackground” which will contain all the graphical elements visible only to maintenance team.

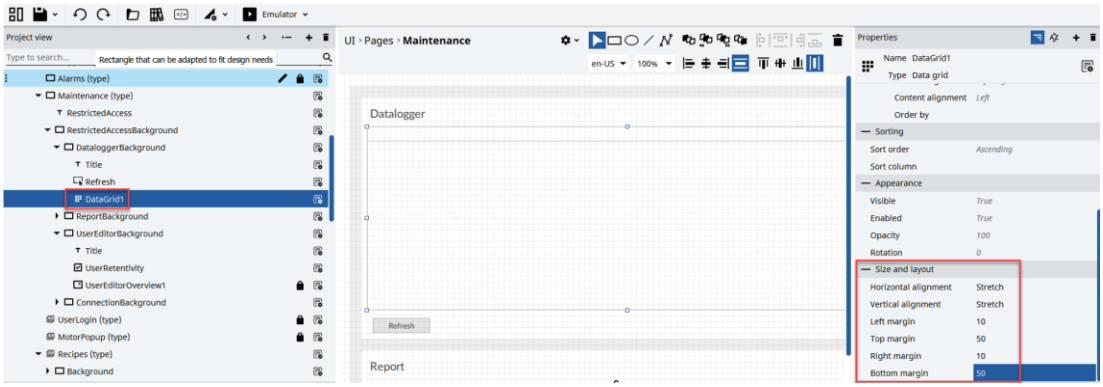
   You will work in the Datalogger square, expand the **DataloggerBackground** and create a new **Data grid**.



2. Adapting the table layout (Scroll down on the properties if you do not see them)

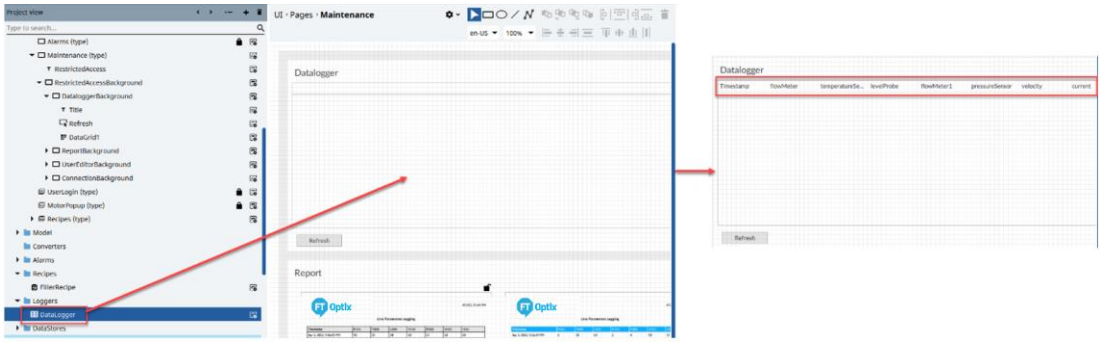
   Adjust the table parameters to:

   - **Horizontal layout:** ***stretch***
   - **Vertical layout:** ***stretch***
   - **Left margin:** ***10px***
   - **Top margin:** ***50px***
   - **Right margin:** ***10px***
   - **Bottom margin:** ***50px***



3. Populating The Data Grid

   Data grid columns are automatically created when dragging a DataLogger to the table, expand the **Loggers** folder, *drag and drop* the **DataLogger** to the table and all the queries and columns will appear in right column of the editor.



4. Refreshing the table content

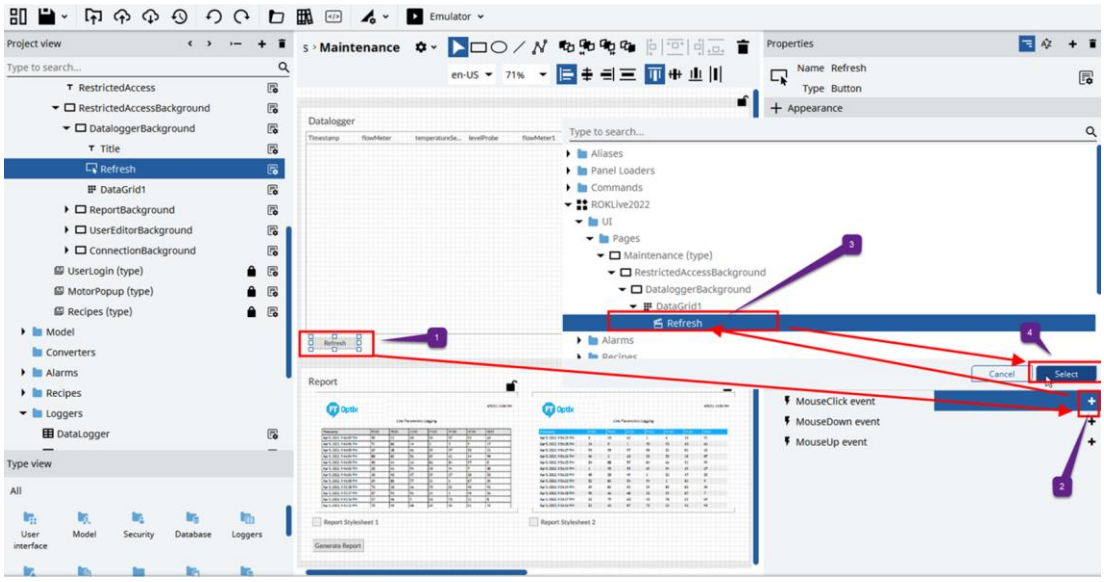
Data grid tables can be refreshed in two ways:

- Automatic: by setting a “Auto refresh time” in the table properties.
- Manual: by adding a button calling the “Refresh” method of the Data grid (leaving the “Auto refresh time” set to zero.

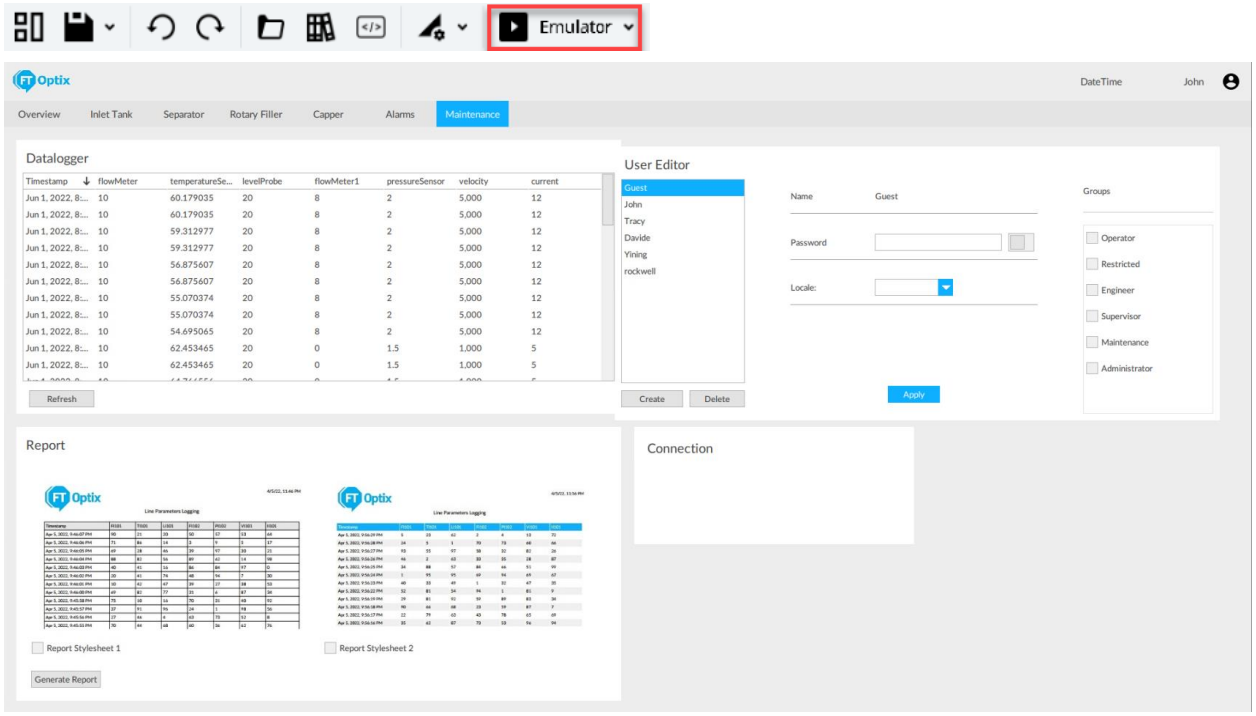
You will now configure the **Refresh** button to update data from the Database, *click* on the **existing** button and add the **Refresh** method, *browse* to

UI > Pages > Maintenance > RestrictedAccessBackground > DataLoggerBackground > DataGrid1

and *select* Refresh.



5. Now run the emulator, Log in as **John** and **click** on the ***Maintenance*** tab.

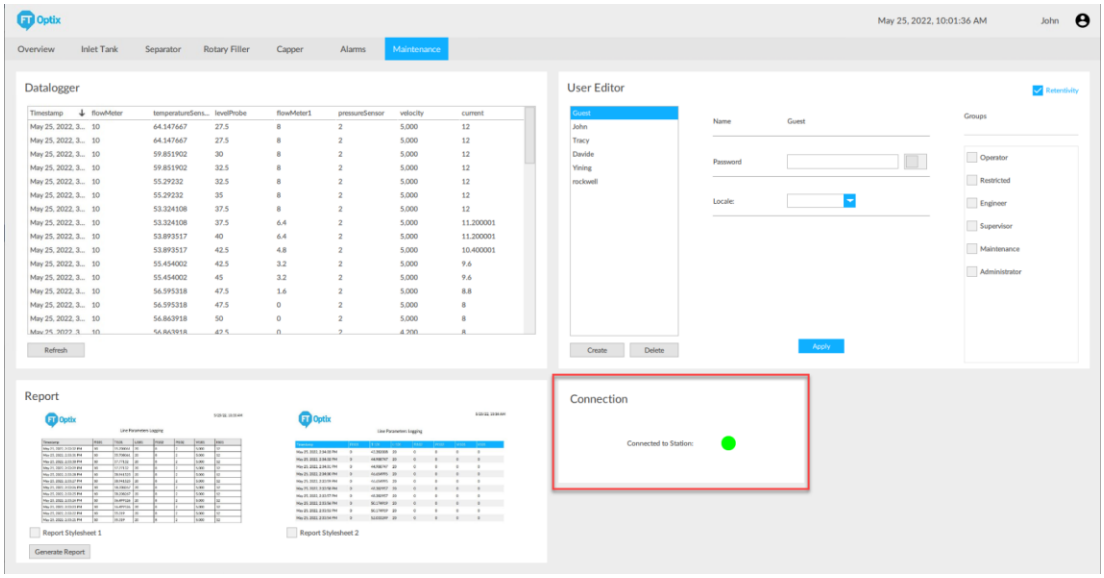


6. Close the **emulator** when you have finished testing.

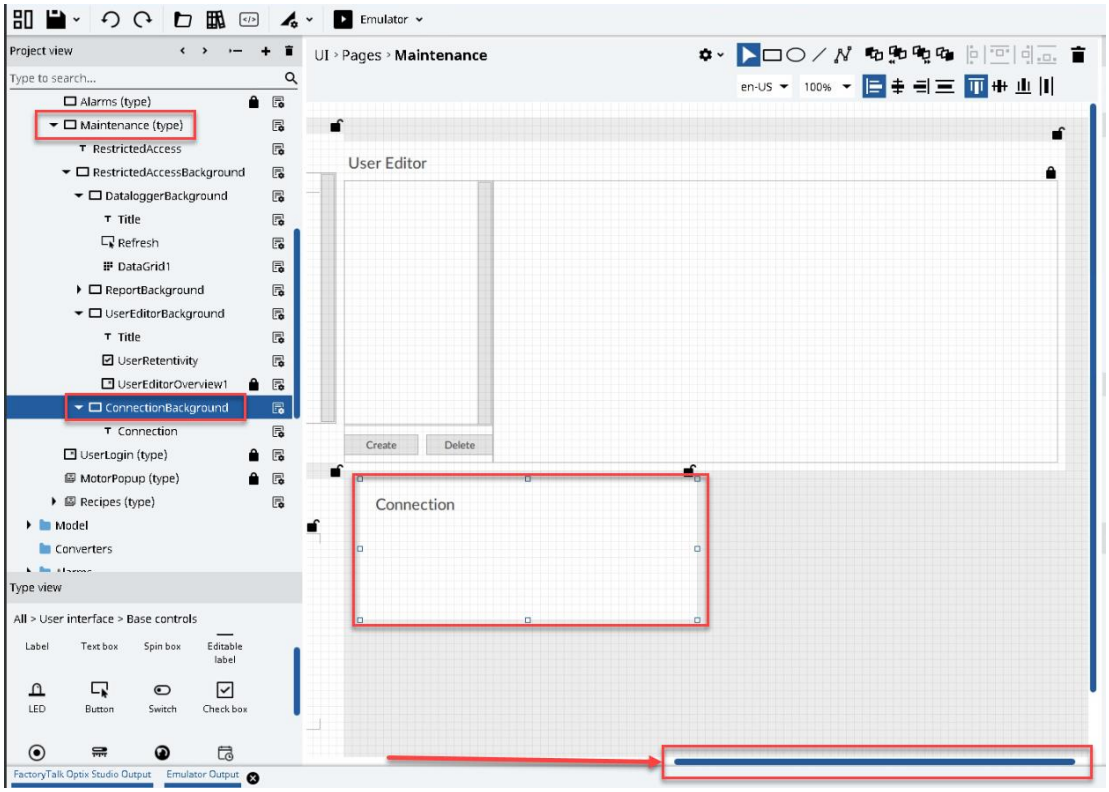
Monitoring the connection status of the controller

In this section of the Lab you will add a Widget to monitor the status of the controller connection.

This will be the result:



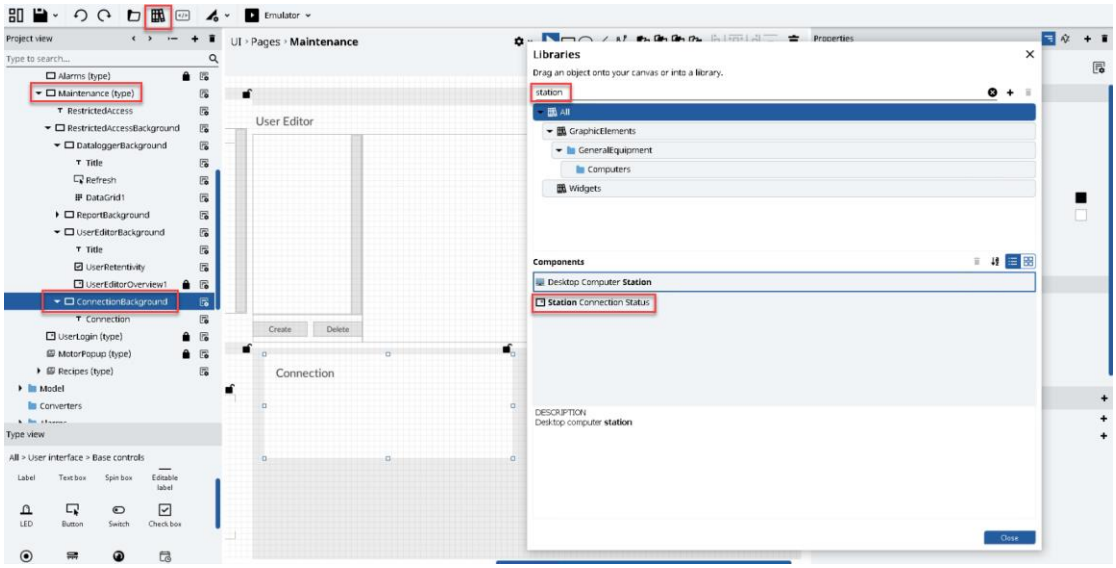
1. Open up the Maintenance screen and scroll to the right so you see the Connection location where we are going to add the Station Connection widget.



2. Adding the Connection Status Widget.

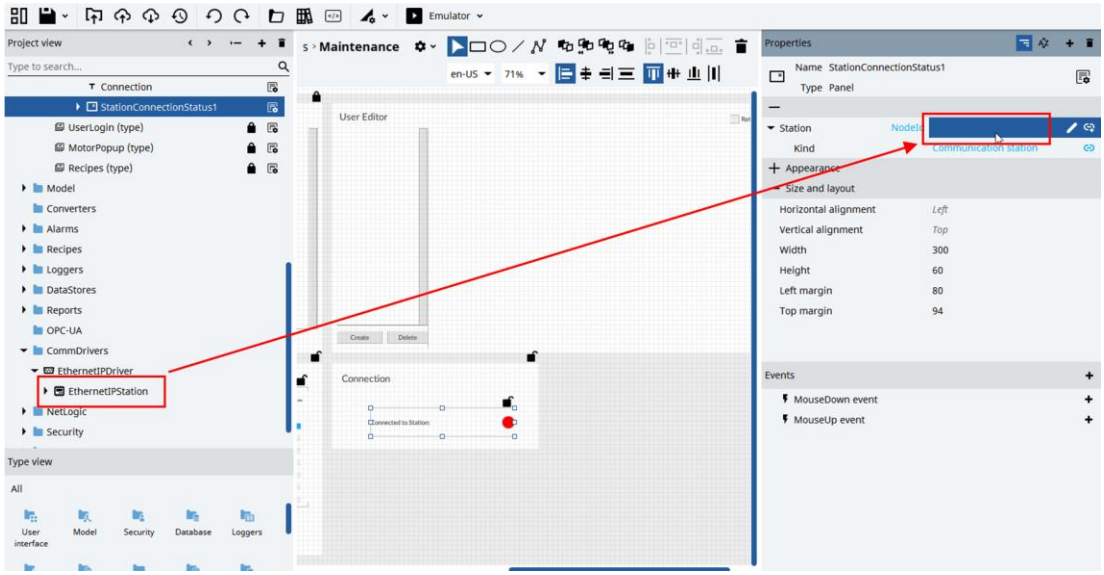
   Using the **Templates Libraries**, you can add a new widget to monitor connection status, it will monitor the real time controller connection and display a red or green LED dependent on the current status.

   Open the **Template Libraries**, type in **station** and *click* **enter** in the **search** bar, *select* **Station Connection Status**, *drag and drop* the **Station Connection** Widget to the **Connection** container.



3. Configuring the widget

   Click on the **StationConnectionStatus1** widget and create a DynamicLink on the **EthernetIpStation** found in **CommDrivers > EthernetIpDriver > EthernetIpStation** by simply *dragging and dropping*.

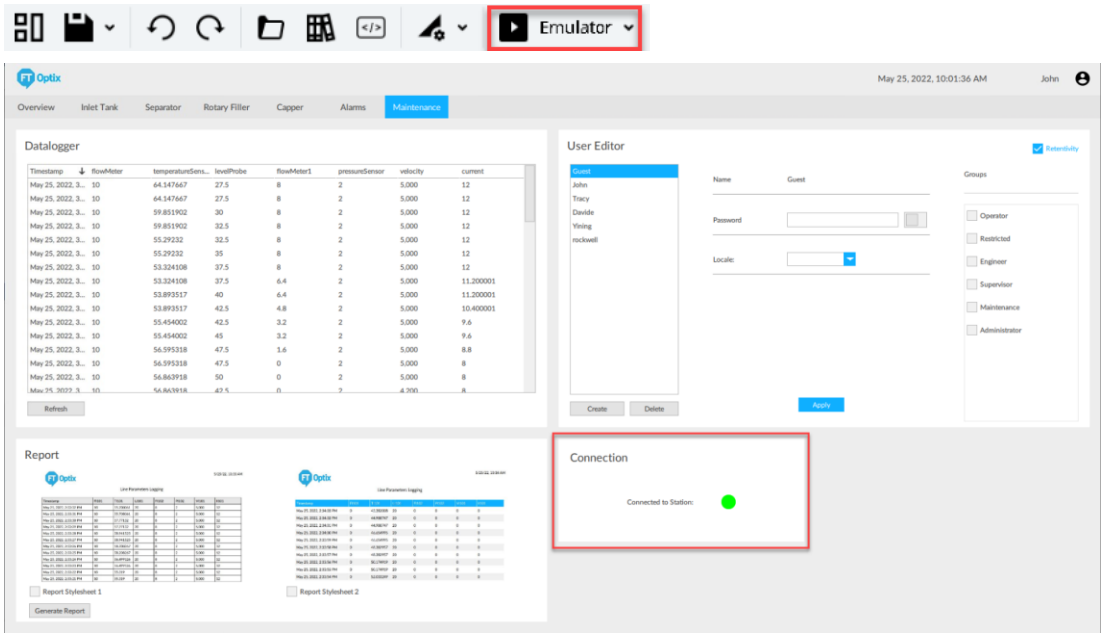


4. Editing the widget layout.

   Click on the widget and using the top icons set alignment to center.

> Objects alignment can be set using the properties column in the right side or using the top icons

Now run the **emulator**, Log in as **John** and **click** on the ***Maintenance*** tab.



4. Close the **emulator** when you have finished testing.

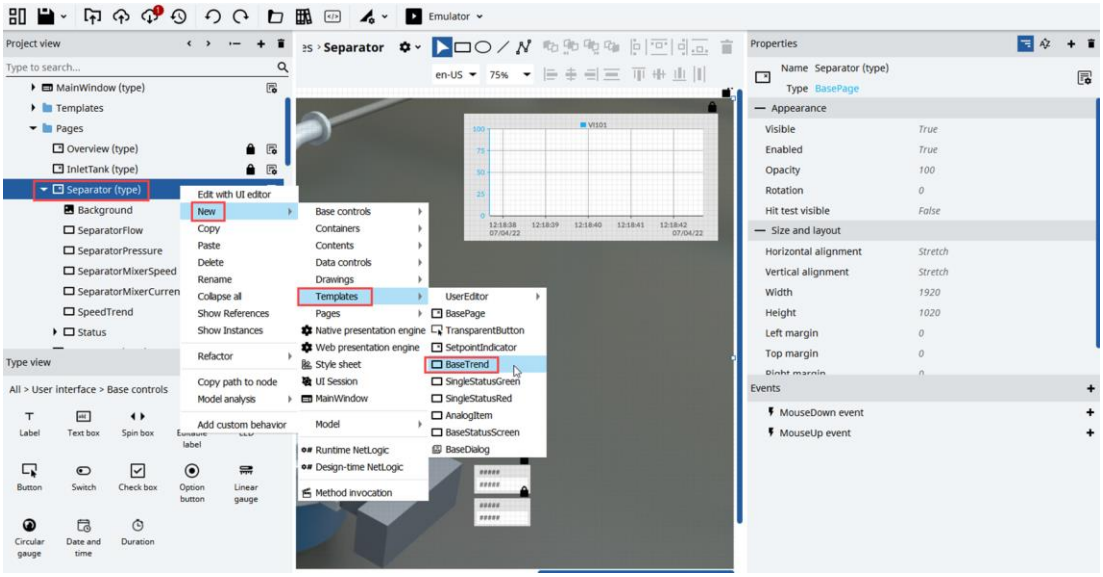
Trending

In this section of the Lab you will learn about trending.

This will be the result:

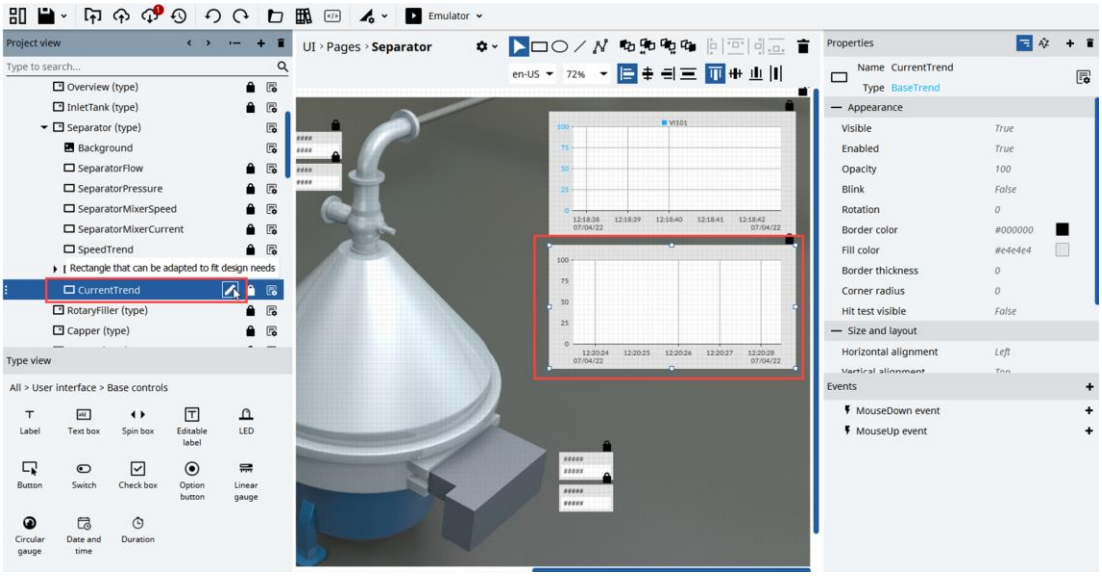
1. Adding a DataTrend to a display.

   Double click on the **Separator** page and create **new instance** of **BaseTrend**.



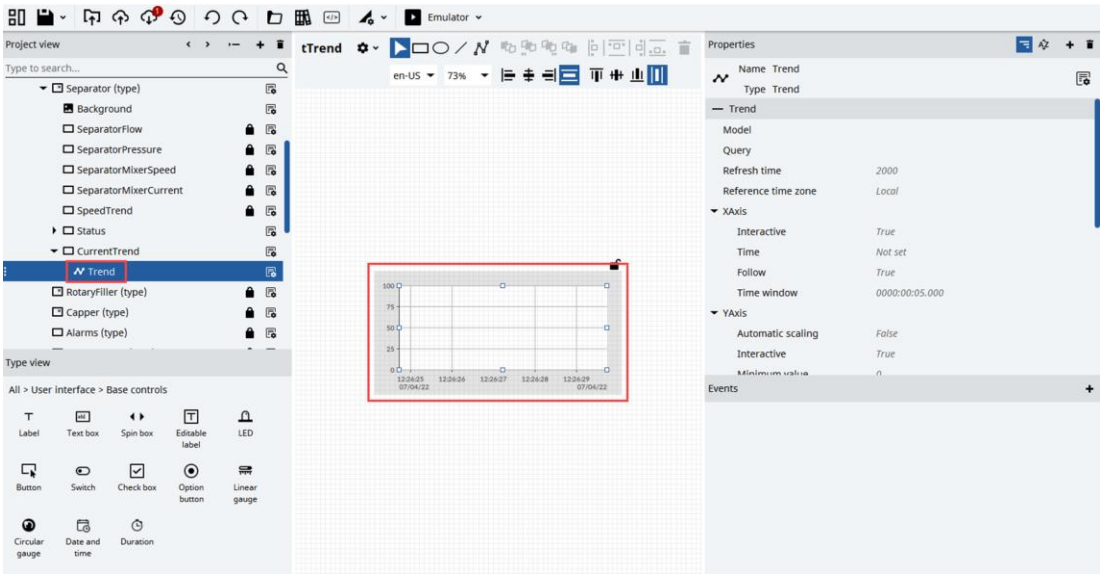
2. Position the new trend

   Drag **BaseTrend** to the right corner of the page and rename it to ***CurrentTrend***.



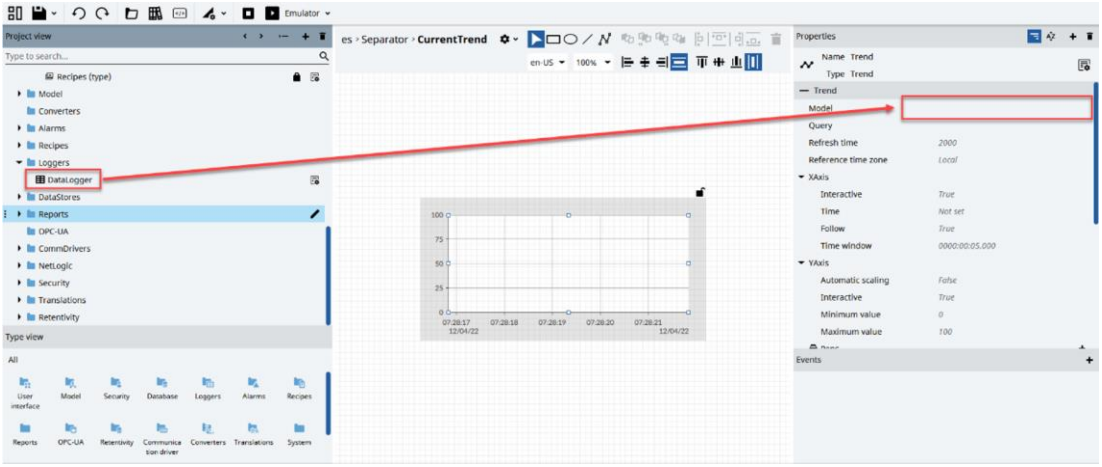
3. Editing the new trend.

   Double click on **CurrentTrend** to open the **editor**.



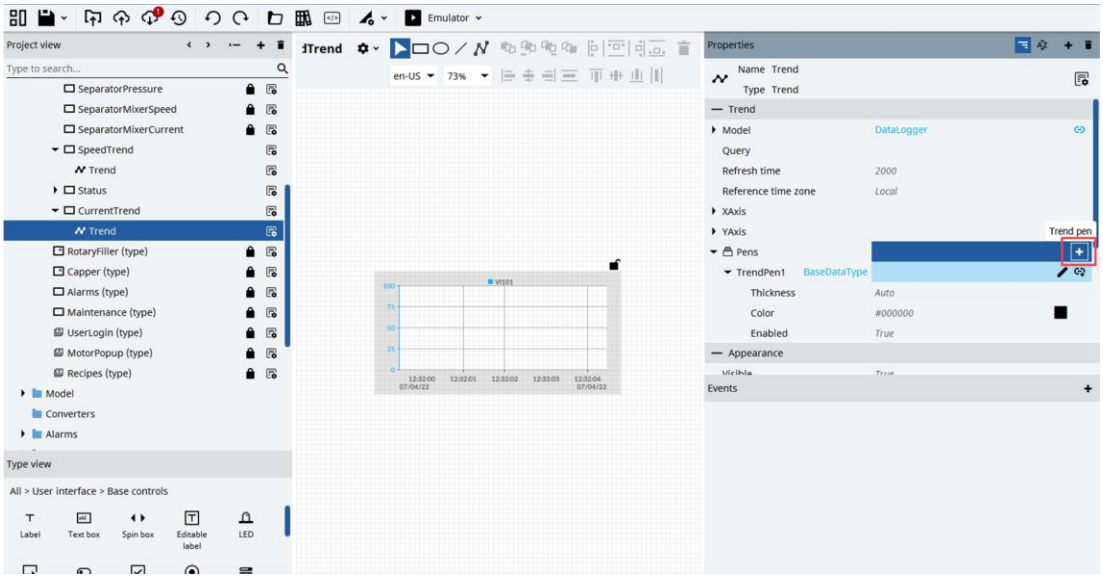
4. Configuring the new trend

   The first step is to link a data source **(DataLogger)** to the **Model** parameter.



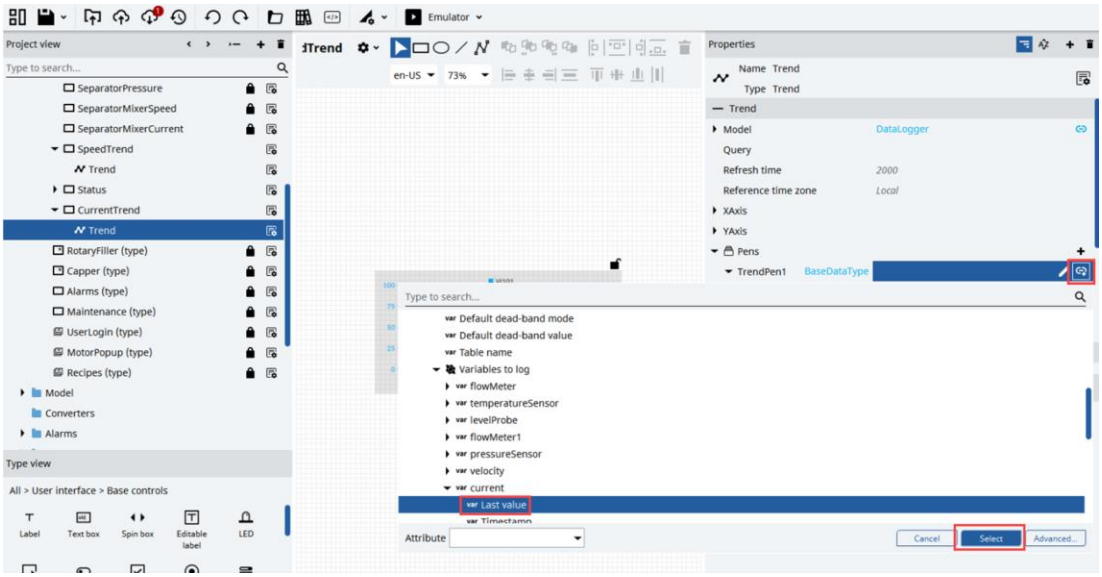
5. Adding pens to the trend.

   Click on the **+** icon in the right part of **Pens** in the **Properties** panel to add a new pen.



6. Configuring a new pen

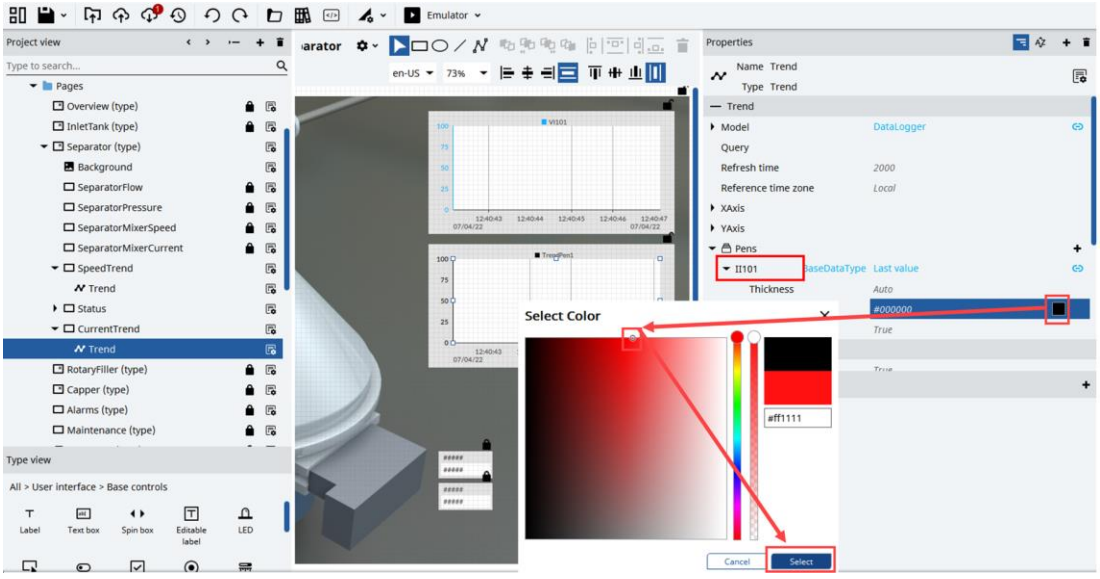
   Using the **add dynamic link** item browse to **Loggers > Datalogger > Variables to log > current > last value** and *click* ***Select***.



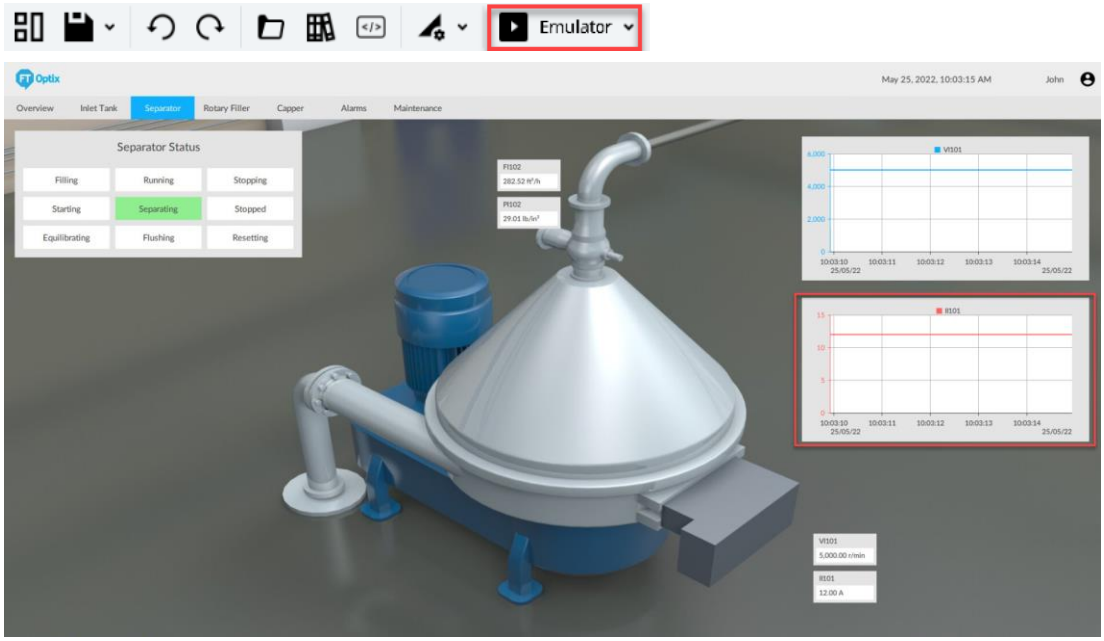
7. Customize the trend pen.

   Click the rename icon to change the name to **II101** and *click* on the color selection icon to pick any color.

> Colors can be both chosen in RGB values (#XXYYZZ) or with standard HTML naming (orange, blue, yellow, …)



8. Now run the **emulator**, click on the Separator tab and you will see the Trend you just added.

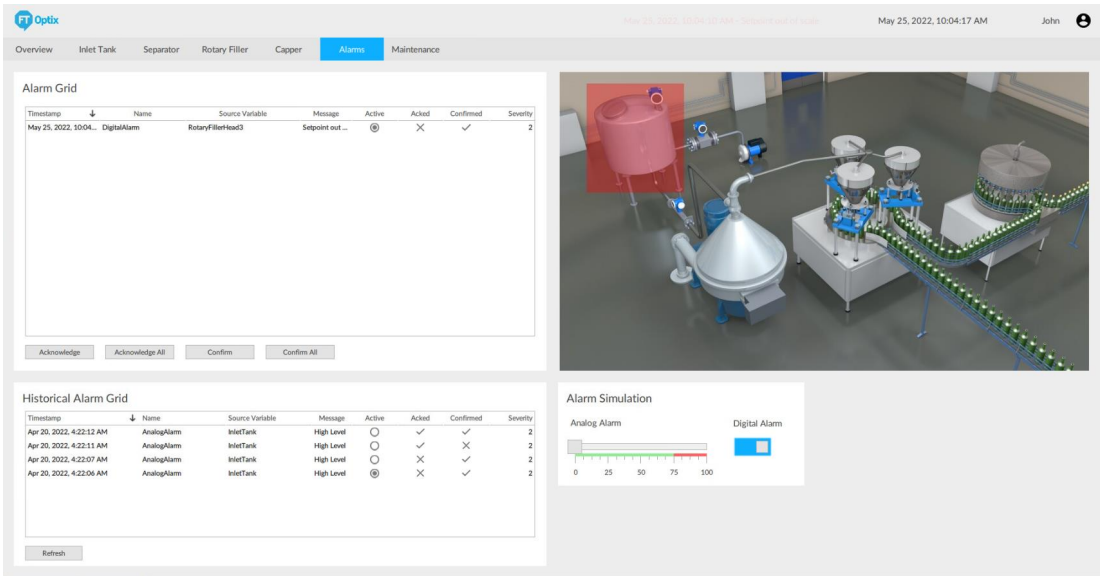


9. Close the **emulator** when you have finished testing.

Alarming

In this section of the Lab you will create an alarm display that contains an Alarm Summary and a Historical Alarms Log Viewer. You will setup the Alarm Datalogger for Historical Alarm Logging and in the header of the project you will create an Alarm Banner.

This will be the result:



Read about FactoryTalk Optix™ Alarms

Introduction

An alarm is a significant event regarding the process or status of the machine, typically linked to anomalies that require the attention or intervention of the operator. An alarm event occurs, and therefore the alarm is activated, particularly when the value of a monitored variable (e.g. a temperature variable, or a variable dedicated to signaling a specific alarm) is different from one or more values set as "normal".

When a PLC with a non-OPC UA server is used, the PLC variables to monitor have to be imported into FactoryTalk Optix™ and referenced in alarm type objects.

On the other hand, if your application communicates with an OPC UA server, FactoryTalk Optix™ can natively communicate with these servers through the OPC UA client object. In particular, it can automatically read the alarm events exposed by the server, their relative statuses, and all other available associated information.

FactoryTalk Optix™ makes it possible, through various objects and widgets, to display alarm data at runtime, to acknowledge them and confirm them.

Alarm types

In FactoryTalk Optix™ Studio, different objects are available for different alarm types. These objects generate an alarm event at runtime when the monitored variables reach the set critical values.

Exclusive and non-exclusive alarms

*All the alarms except digital alarms are called **Limit alarms**. In these alarms, it is possible to set up to four limits through the **Low-low limit**, **Low limit**, **High limit** and **High-high limit** properties. Depending on the alarm type, these limits determine values or ranges of values in which the monitored variable can be found. The alarm can signal one or more limits reached:*

Type	Description	Application example
Digital alarm	Monitoring of Boolean variables. It activates when the value of the monitored variable is not the normal value.	Monitoring of a variable for the open/closed status of a firewall. If the "closed" normal condition is not met, an alarm event is generated.
Exclusive level alarm Non-exclusive level alarm	Monitor the reaching of values of interest for variables relative to physical quantities. It activates when the value of the monitored variable reaches one or more limits of interest.	Monitoring of a temperature variable. For example, if the "low temperature" limit is set to 10 and the "high temperature" limit is set to 100 on a scale from 0 to 100 (°C), when the temperature reaches these limits an alarm event is generated.
Exclusive deviation alarm Non-exclusive deviation alarm	Monitor the deviation from a setpoint value for variables relative to physical quantities. It activates when the value of the monitored variable has an offset equal to or greater than the one set.	Monitoring of a variable relative to a temperature that must be kept constant. If a setpoint value is set to 50 (°C), "temperature drop" limit set to -3, and "temperature increase" limit set to +3, when the temperature reaches 47 °C or 53 °C, an alarm event is generated.
Exclusive rate of change alarm Non-exclusive rate of change alarm	Monitor the rate of change of the value of a variable in a set time. It activates when the rate of change of the value of the monitored variable reaches one or more set limits.	Monitoring of a variable relative to the speed of a conveyor belt. If the sampling time is set to 5 seconds and the change limit set to 10 (m/s), and if the rate of change of the value is equal to or greater than 10 with respect to the previous sample value in the 5 seconds, an alarm event is generated.
Non-exclusive	When several limits of interest are reached consecutively, the alarm signals all the corresponding levels.	If a temperature, starting from a normal value, reaches the high-high level, the alarm signals both the high-high and high level.
Exclusive	When several limits of interest are reached consecutively, the alarm signals only the last level reached.	If a temperature, starting from a normal value, reaches the high-high level, the alarm signals only the high-high level.

Native widgets to view and manage alarms

*The **Alarm grid** widget makes it possible to display and manage the alarms at runtime. View the information on alarm events (timestamp, name, source variable, message, severity) and its statuses (active, acknowledged, confirmed) in a table and allow the user to acknowledge and confirm one or all the alarms with dedicated buttons.*

*The **Alarm log grid** widget displays the log of alarm events recorded by a suitably configured event logger (library object **Alarm events logger**) in a database. View the information on alarm events (timestamp, name, source variable, message, severity) and its statuses (active, acknowledged, confirmed) in a table. Every row in the table corresponds to a change in status of the alarm event.*

*The **Alarm log grid with filter** widget displays the logged data in the same way and makes it possible to filter them temporally.*

Custom alarm types and instances

To customize native alarm types with additional properties, custom alarm types can be configured. Typically, custom alarms are configured to display columns with additional information in the Alarm Grid, for example, the name of the mechanical system affected by the alarm, or to be able to filter the alarms in the grid according to a specific criterion.

Creating custom alarm types also guarantees efficiency and consistency, since it is possible to derive their instances.

Example

Let us consider that there is a machine with several temperature detectors.

*Instead of configuring individual alarms for the different detectors, in FactoryTalk Optix™ it is possible to configure the **TemperatureAlarm** alarm type with a custom **DetectorID** field specific for the ID of the detector connected to the monitored variable.*

*By creating instances of **TemperatureAlarm** in the Alarm Grid, it is possible to distinguish the **DetectorID** at the origin of the alarms and filter the alarms in the grid according to a specific value of **DetectorID**.*

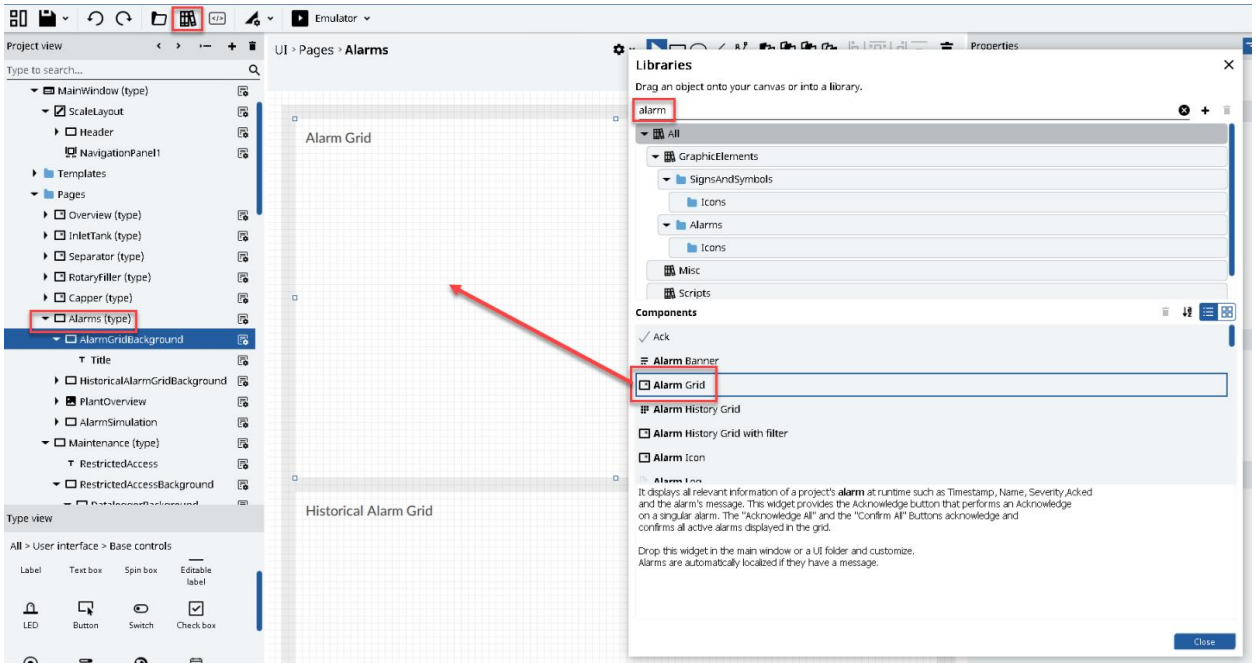
Branching enabled

At runtime, if the same alarm activates repeatedly, by default the related alarm object in FactoryTalk Optix™ only signals the most recent event and displays it in the Alarms grid widget. The alarm must then be acknowledged and confirmed. Disappears if the relative variable returns to its normal status in the PLC.

However, using the Alarm branching property of the project node, it is possible to enable the signaling of multiple active alarms for a same alarm object. In the Alarms grid widget, all the active alarms for a same alarm object are then displayed, which must all be acknowledged and confirmed. The alarms stop if the related variables return to normal status on the PLC.

1. Adding the Alarm Summary DataGrid.

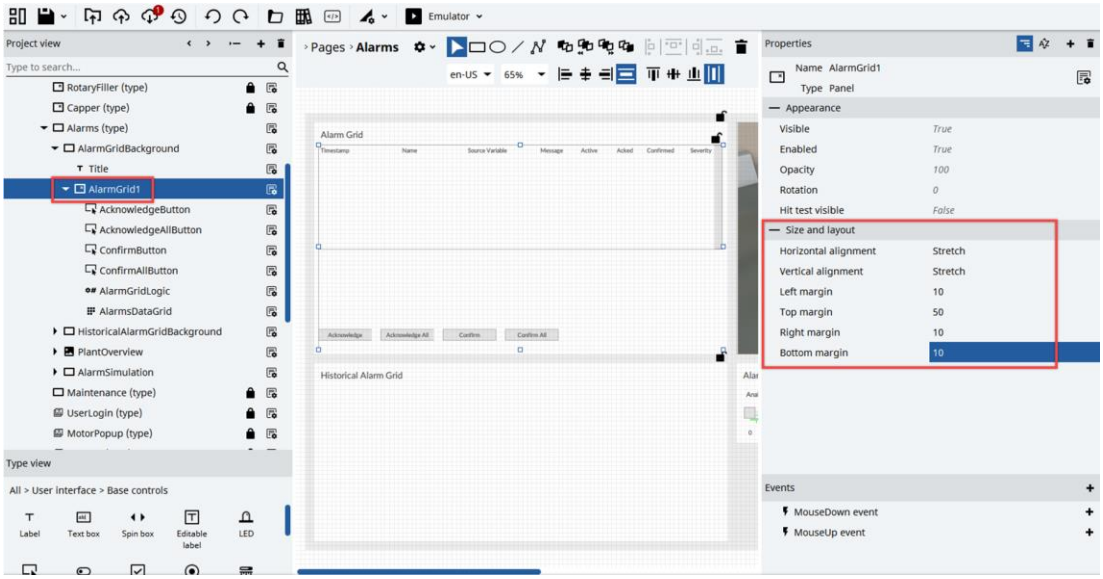
   Double click on the **Alarms** page, here you will add the Alarm Summary DataGrid that shows real time alarms. Open the **Templates Libraries** and locate the **Alarm Grid** widget, *drag and drop* this widget to the top left white container.



2. Customizing the Alarms DataGrid

Click on **AlarmGrid1** and change its **Properties** to:

- **Horizontal alignment:** ***stretch***
- **Vertical alignment:** ***stretch***
- **Left margin:** ***10px***
- **Top margin:** ***50px***
- **Right margin:** ***10px***
- **Bottom margin:** ***10px***



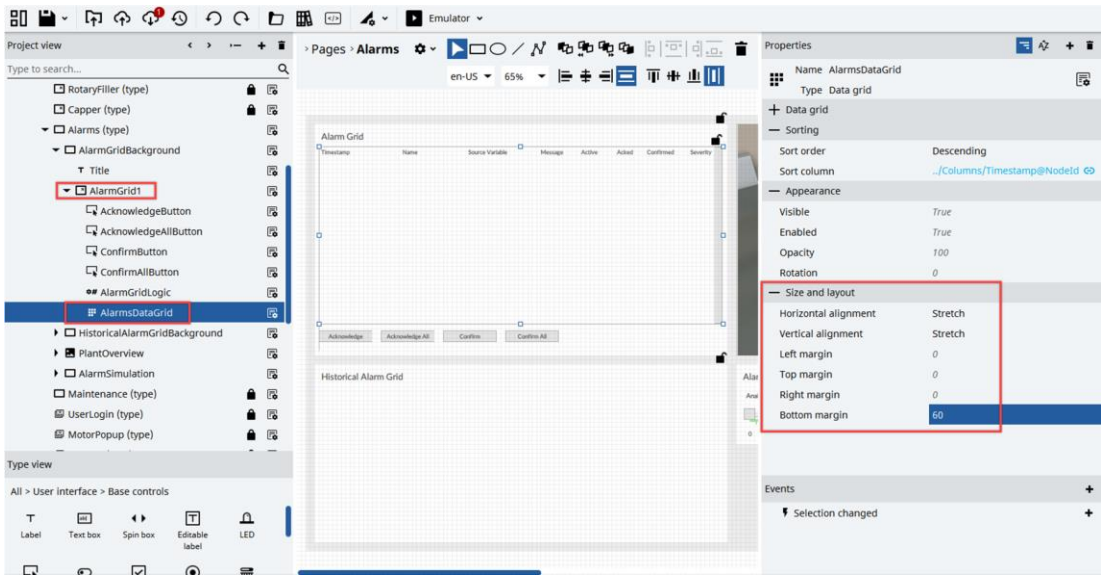
3. Further widget customization

Every widget is composed of multiple base items that can be arranged as preferred; you are now going to stretch the viewable Alarm DataGrid to fill most of the container.

Expand the **AlarmGrid1** widget and *select* **AlarmsDataGrid**, set its properties to:

- **Horizontal alignment:** ***stretch***
- **Vertical alignment:** ***stretch***
- **Bottom margin:** ***60px***

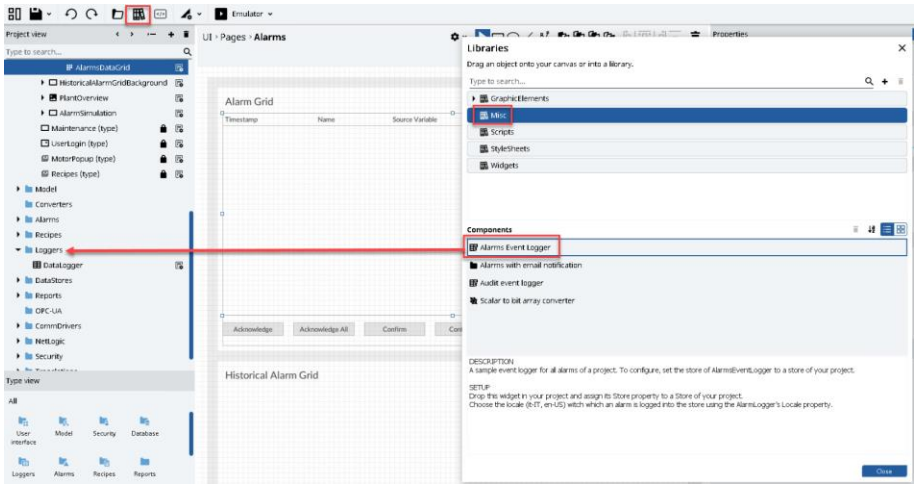
> *Every single item in the Template Library can be completely customized, they can be used as starting templates to create custom elements or used as-it-is.*



Now we will take a look at alarm history.

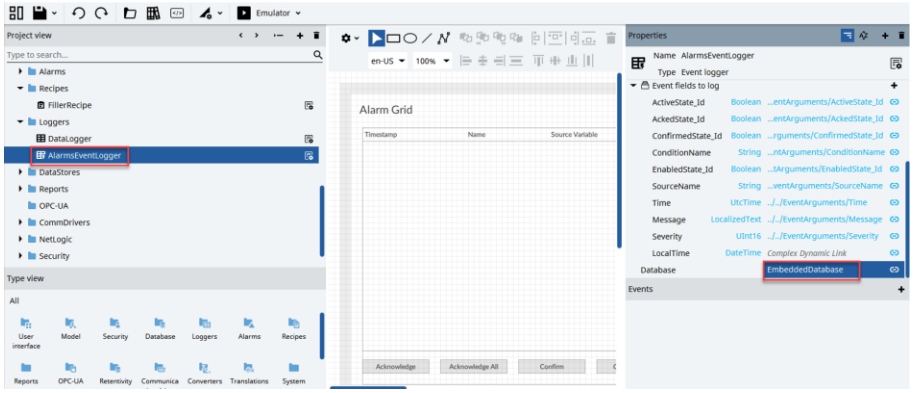
4. Adding the alarm logger.

   Open the **Templates Libraries**, go to folder **Misc** and *drag* the **Alarms Event Logger** to the **Loggers** folder of the project. Rename **AlarmsEventLogger1** to **AlarmsEventLogger**.



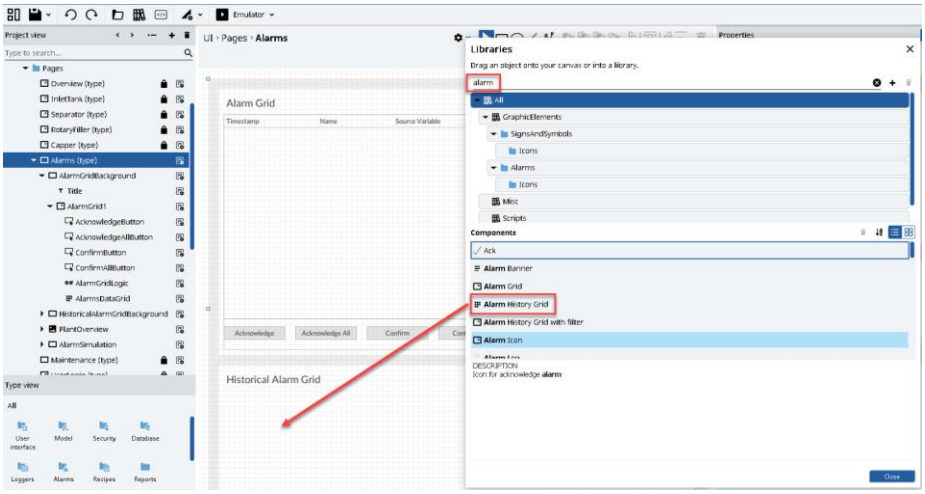
5. Configure the Alarm Event Logger.

   Click on the **AlarmsEventLogger** and edit the last parameter to set the database you are going to use, *click* the **EmbeddedDatabase**.



6. Add Historical Alarm Grid to the project.

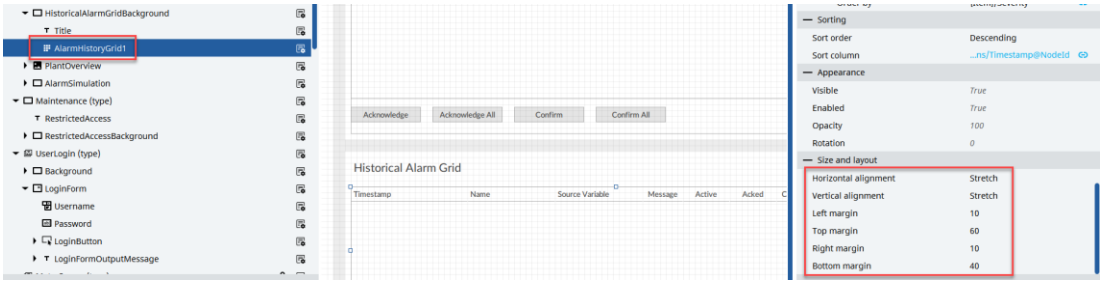
   Open the **Alarms** display, open the **Template Libraries**, type in **Alarm** in the search bar and drag the **Alarm History Grid** in the bottom left block of the page.



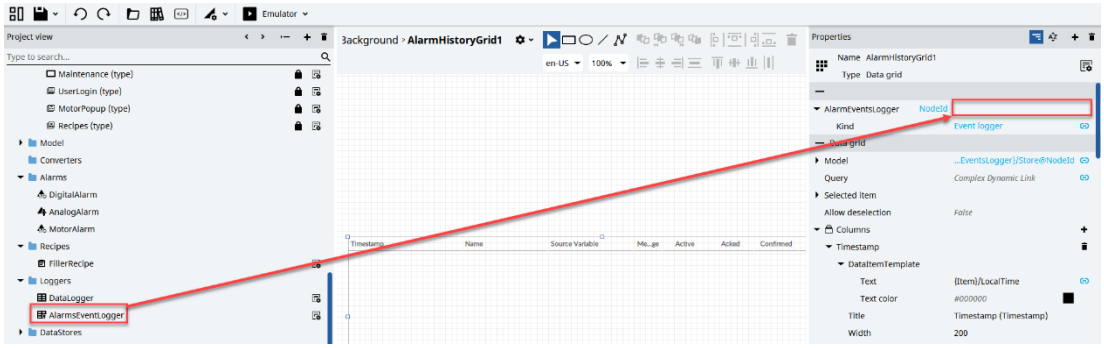
7. Customize Historical Alarm Grid

   Click on AlarmHistoryGrid1 and set its parameters:

   - **Horizontal alignment:** ***stretch***
   - **Vertical alignment:** ***stretch***
   - **Left margin:** ***10px***
   - **Top margin:** ***60px***
   - **Right margin:** ***10px***
   - **Bottom margin:** ***40px***

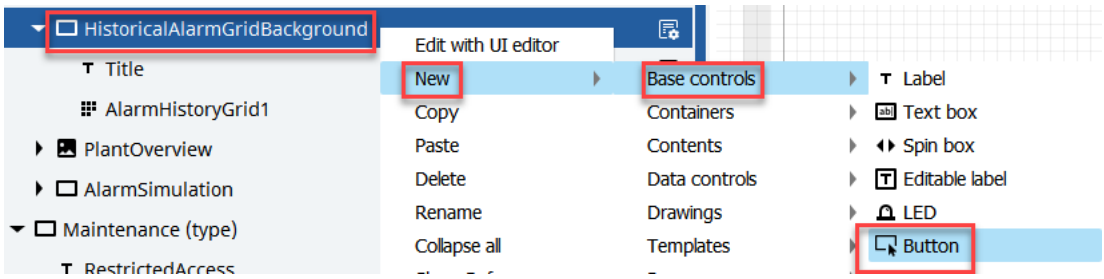


8. *Drag and drop* the **AlarmsEventLogger** to the **AlarmsEventLogger** parameter in the **AlarmHistoryGrid1**.



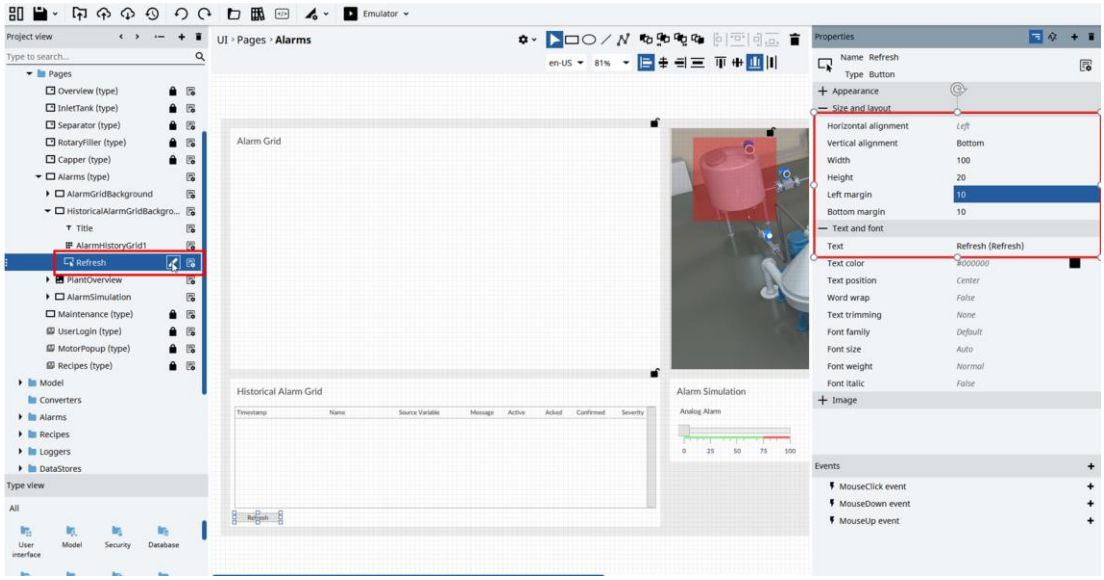
9. Refresh HistoricalAlarmGrid

   Create new button inside **HistoricalAlarmGridBackground** container,

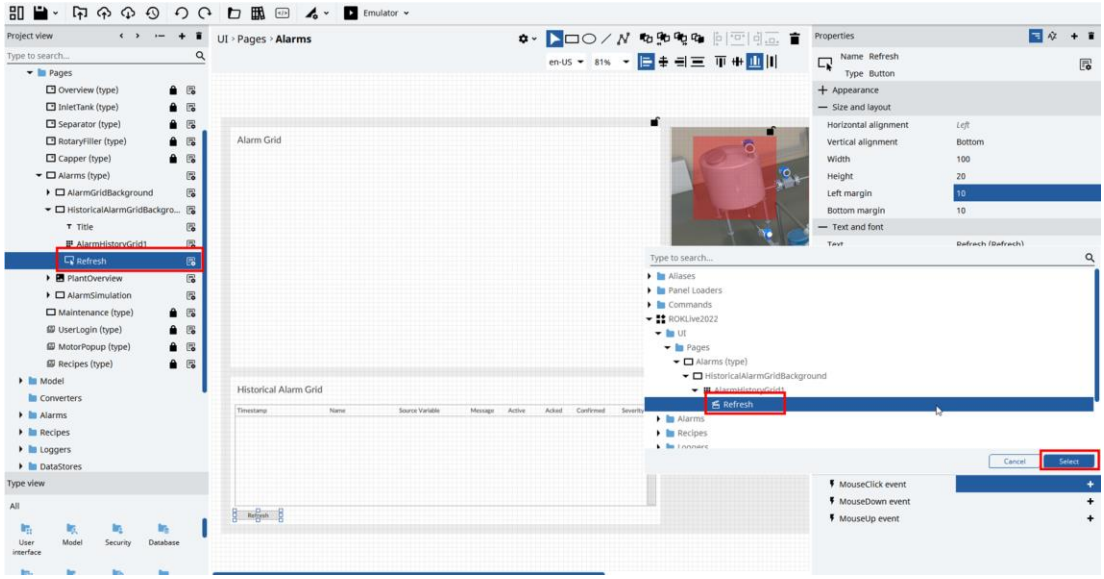


10. Rename **Button1** to **Refresh** and set properties to:

- **Vertical alignment:** **Bottom**
- **Width:** **100px**
- **Height:** **20px**
- **Left margin:** **10px**
- **Bottom margin:** **10px**
- **Text:** **Refresh (choose existent translation and hit enter)**

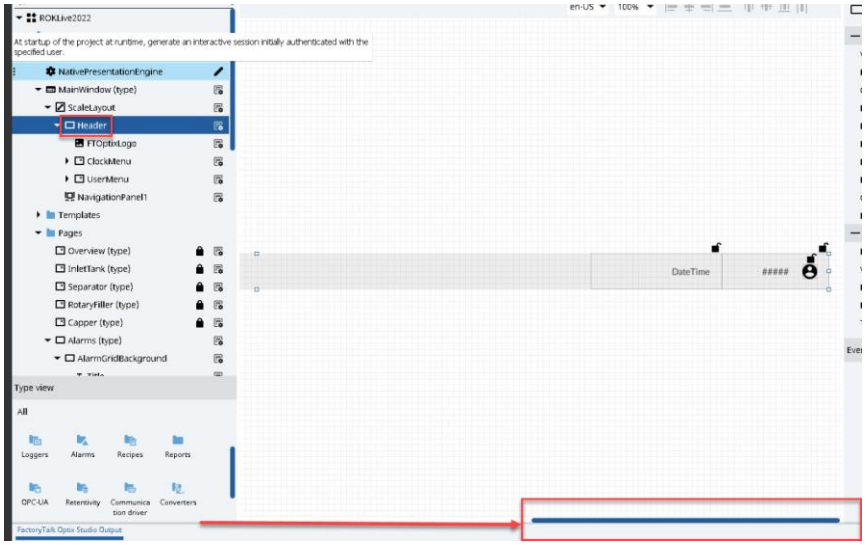


11. Configure Refresh event. Click on the "**+**" icon close to MouseDown event and browse to **ROKLive2022 > UI > Pages > Alarms > HistoricalAlarmGridBackground > AlarmHistoryGrid1** and *select* **Refresh** method

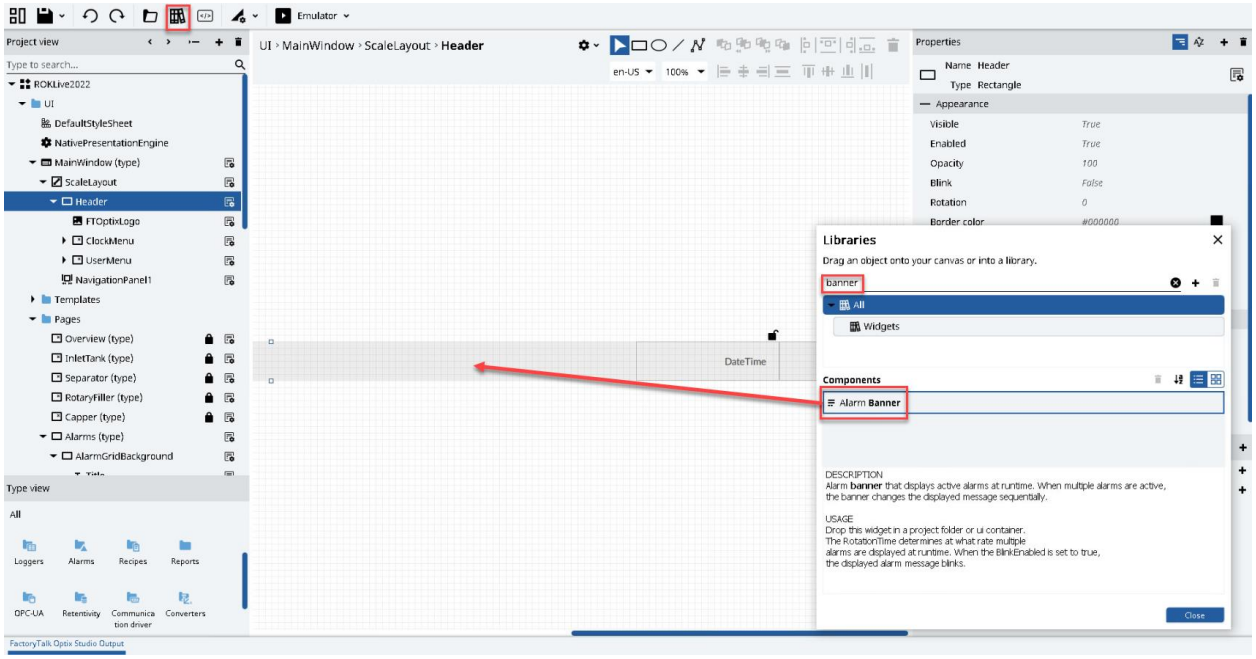


12. Insert the Alarm Banner.

    Double click on **Header** in **MainWindows > ScaleLayout > Header** and move the bar to the right.



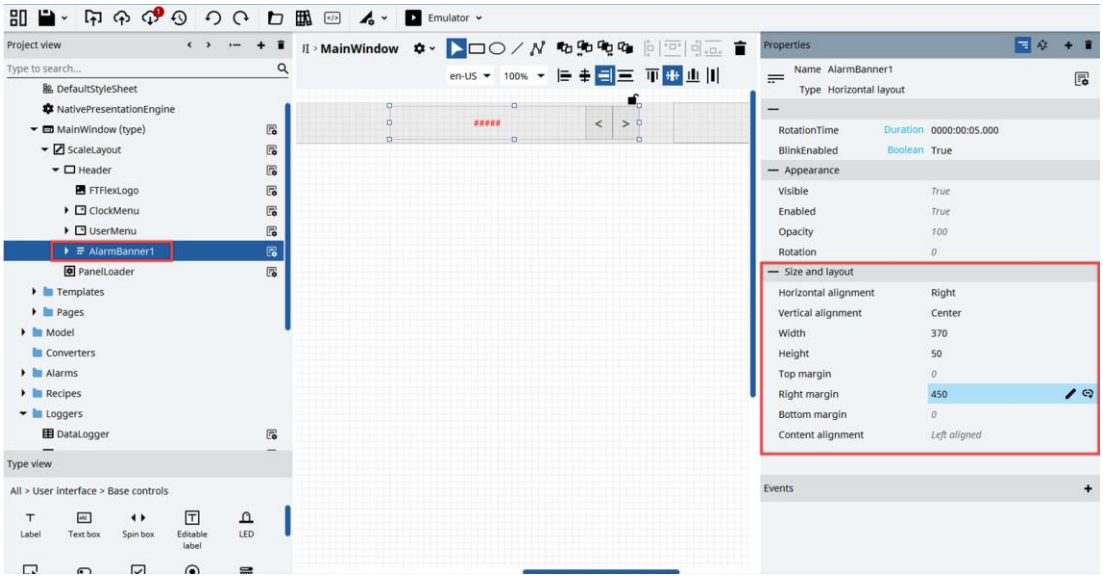
13. Open the **Templates Libraries**, type in **banner** in the search bar and drag the Alarm Banner to the Header.



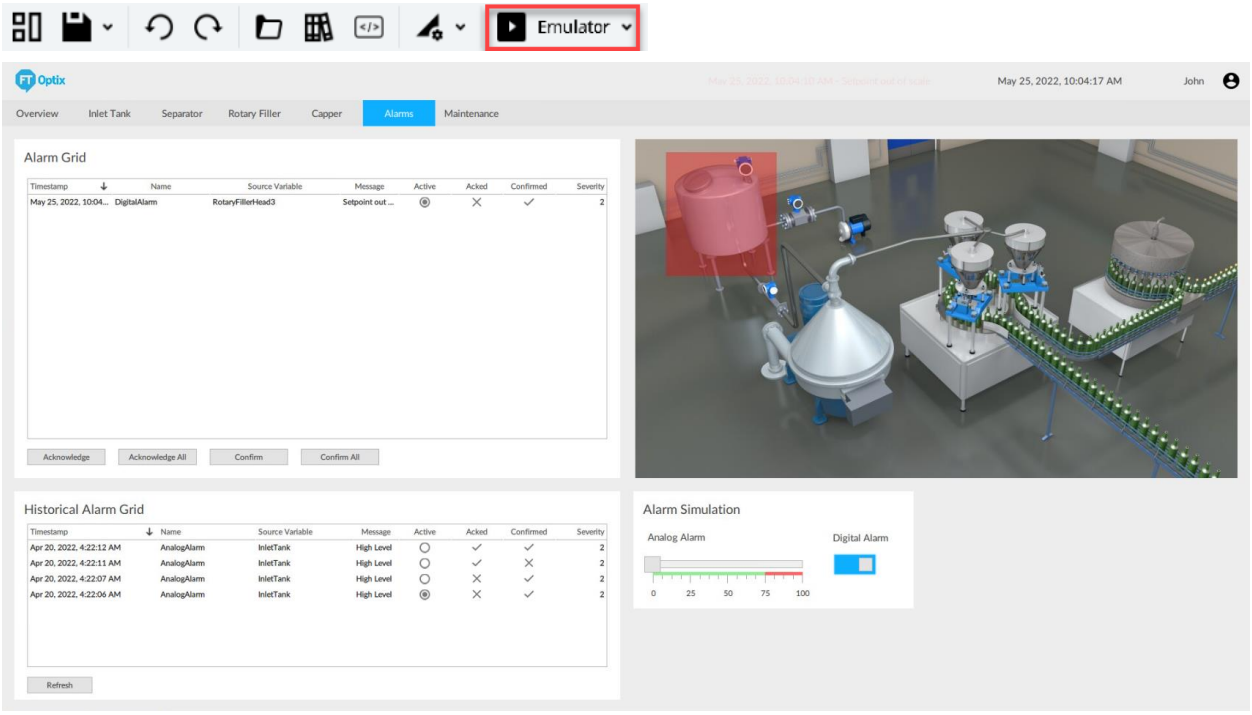
14. Customizing the Alarm Banner.

    Click new **AlarmBanner1** and set the following **Properties** to:

    - **Horizontal alignment:** ***right***
    - **Vertical alignment:** ***center***
    - **Right margin:** ***450***



15. Now run the **emulator**, click on the alarms tab and investigate the Runtime Alarming functionality by activating Digital and analog alarms with the **Alarm Simulation**.

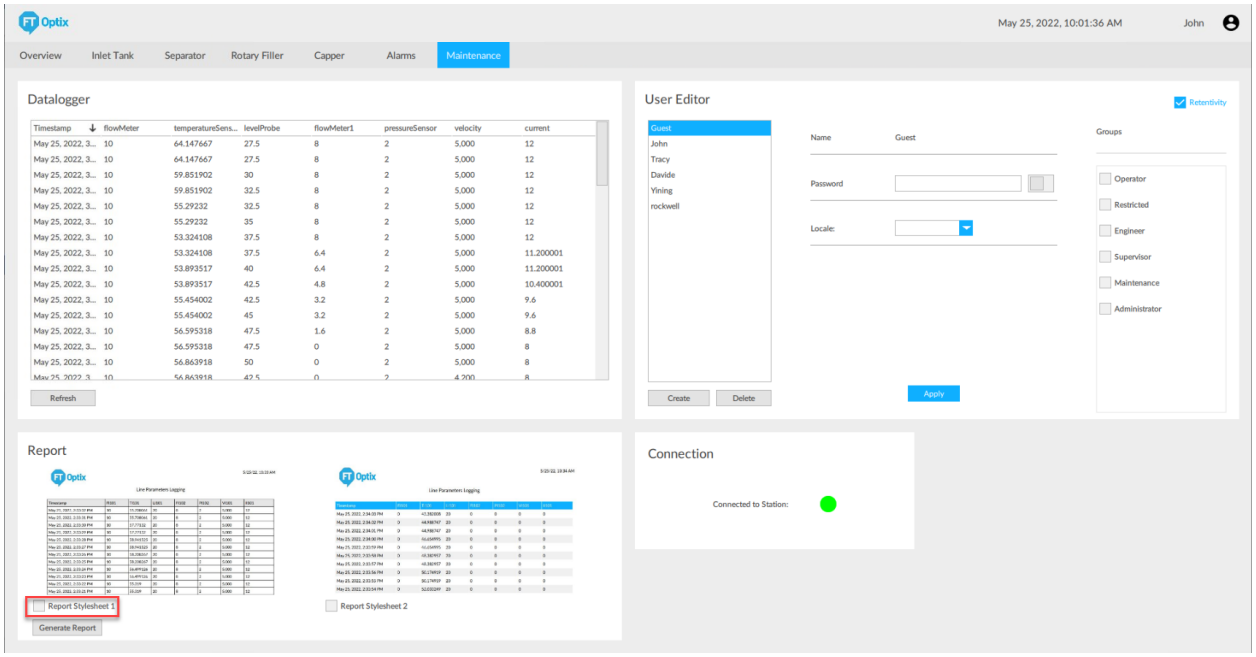


16. Close the **emulator** when you have finished testing.

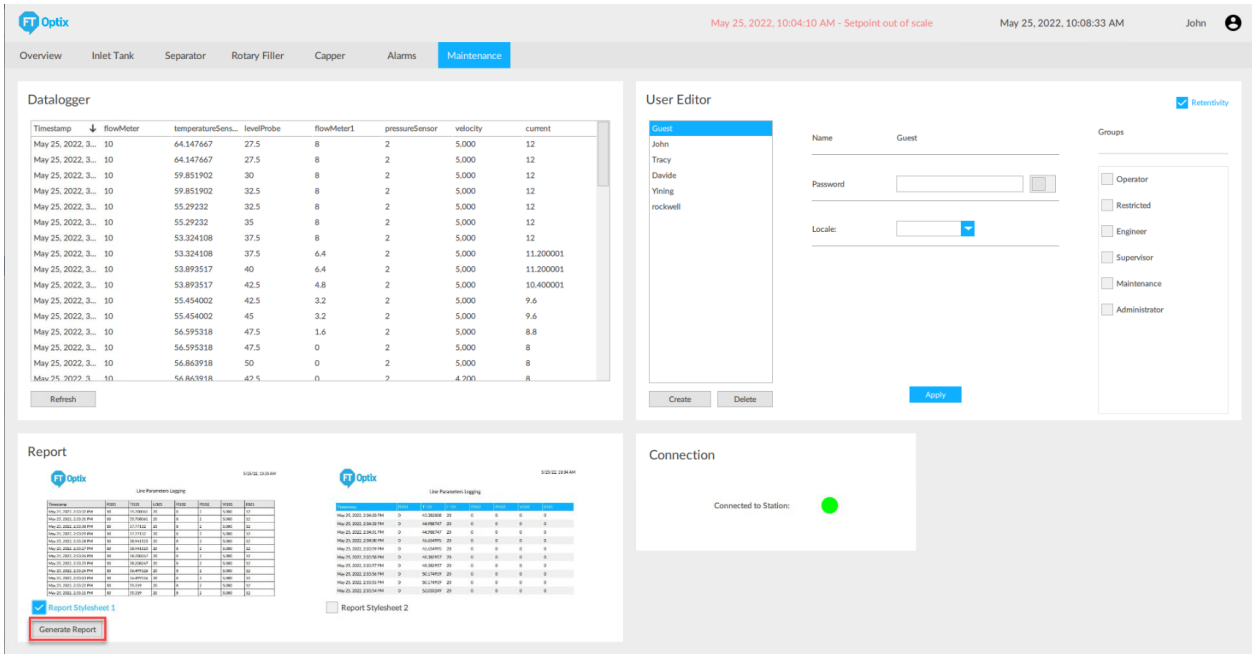
Reporting

In this section of the Lab you will see that pdf reports can be generated in custom styles in Runtime.

1. Start the **Emulator**, Log in as John and *click* on the **Maintenance** tab.
2. Check the **Report Stylesheet1** checkbox.

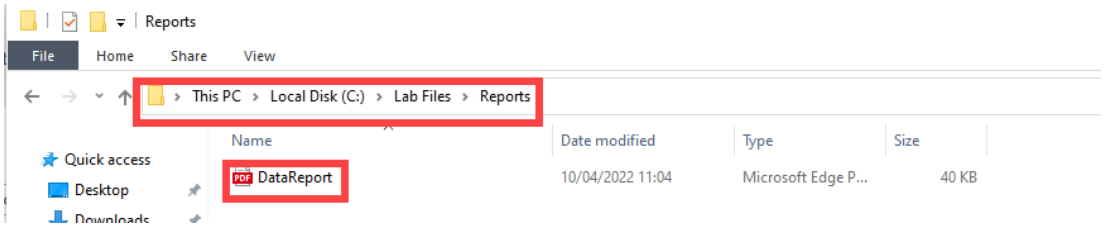


3. Click on ***Generate Report***.

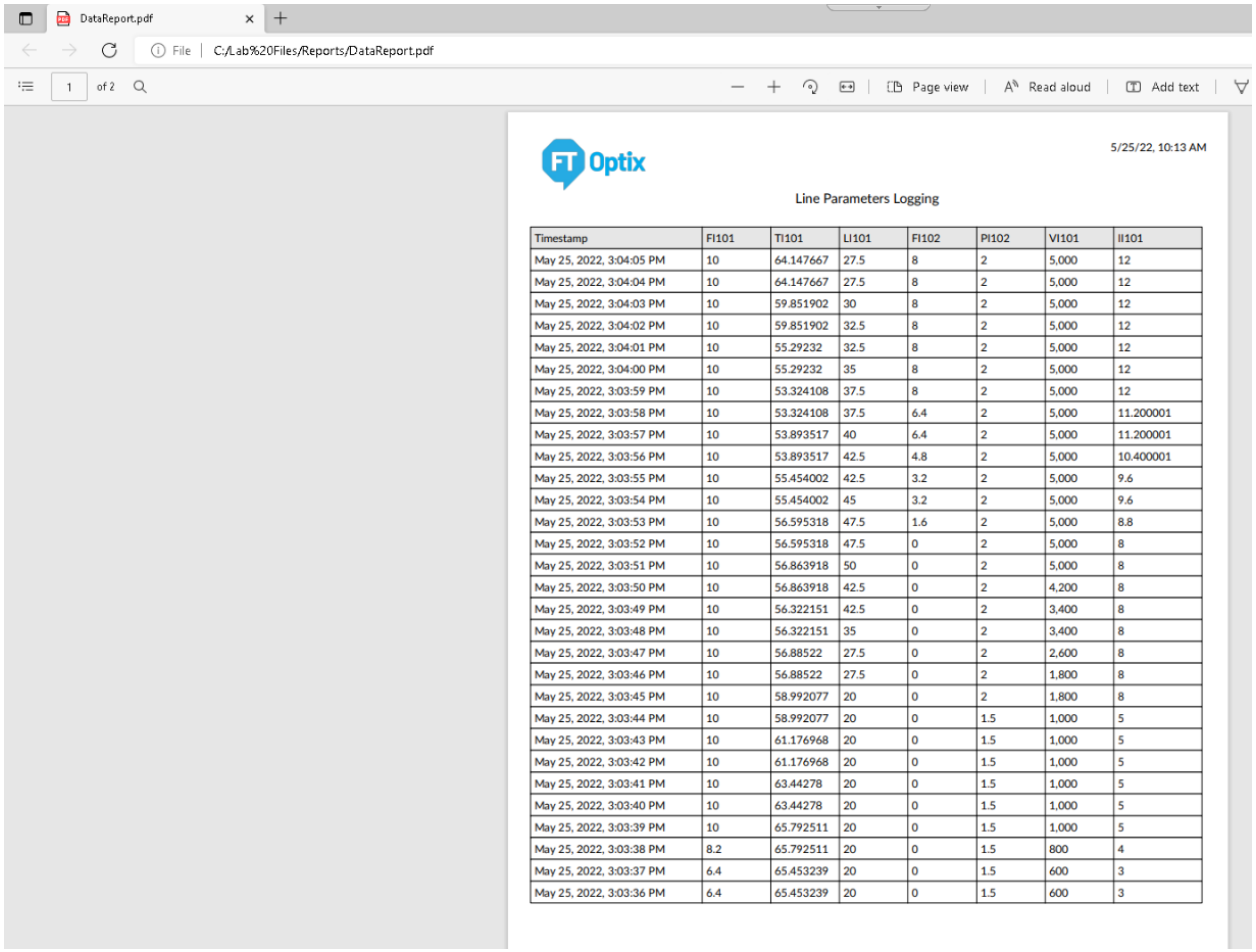


4. Open the folder **C:\Lab Files\Reports**.

   Double click on the **DataReport** pdf.



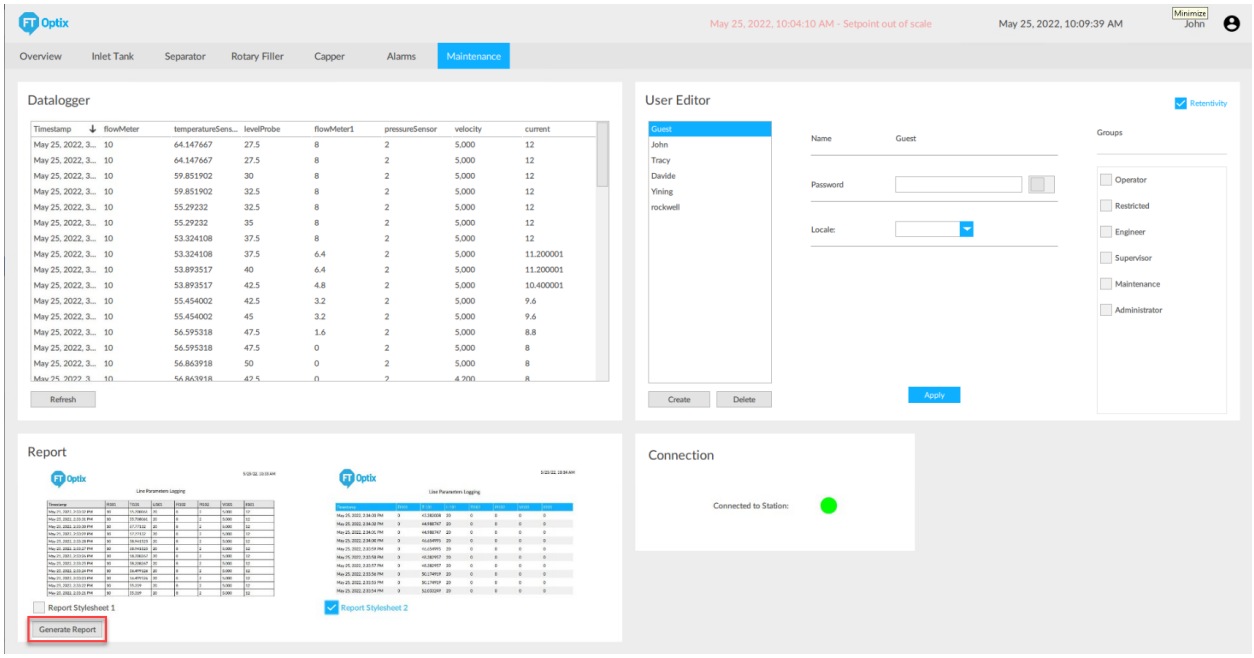
5. The pdf report will open in MS Edge.



Timestamp	FI101	TI101	LI101	FI102	PI102	VI101	II101
May 25, 2022, 3:04:05 PM	10	64.147667	27.5	8	2	5,000	12
May 25, 2022, 3:04:04 PM	10	64.147667	27.5	8	2	5,000	12
May 25, 2022, 3:04:03 PM	10	59.851902	30	8	2	5,000	12
May 25, 2022, 3:04:02 PM	10	59.851902	32.5	8	2	5,000	12
May 25, 2022, 3:04:01 PM	10	55.29232	32.5	8	2	5,000	12
May 25, 2022, 3:04:00 PM	10	55.29232	35	8	2	5,000	12
May 25, 2022, 3:03:59 PM	10	53.324108	37.5	8	2	5,000	12
May 25, 2022, 3:03:58 PM	10	53.324108	37.5	6.4	2	5,000	11.200001
May 25, 2022, 3:03:57 PM	10	53.893517	40	6.4	2	5,000	11.200001
May 25, 2022, 3:03:56 PM	10	53.893517	42.5	4.8	2	5,000	10.400001
May 25, 2022, 3:03:55 PM	10	55.454002	42.5	3.2	2	5,000	9.6
May 25, 2022, 3:03:54 PM	10	55.454002	45	3.2	2	5,000	9.6
May 25, 2022, 3:03:53 PM	10	56.595318	47.5	1.6	2	5,000	8.8
May 25, 2022, 3:03:52 PM	10	56.595318	47.5	0	2	5,000	8
May 25, 2022, 3:03:51 PM	10	56.863918	50	0	2	5,000	8
May 25, 2022, 3:03:50 PM	10	56.863918	42.5	0	2	4,200	8
May 25, 2022, 3:03:49 PM	10	56.322151	42.5	0	2	3,400	8
May 25, 2022, 3:03:48 PM	10	56.322151	35	0	2	3,400	8
May 25, 2022, 3:03:47 PM	10	56.88522	27.5	0	2	2,600	8
May 25, 2022, 3:03:46 PM	10	56.88522	27.5	0	2	1,800	8
May 25, 2022, 3:03:45 PM	10	58.992077	20	0	2	1,800	8
May 25, 2022, 3:03:44 PM	10	58.992077	20	0	1.5	1,000	5
May 25, 2022, 3:03:43 PM	10	61.176968	20	0	1.5	1,000	5
May 25, 2022, 3:03:42 PM	10	61.176968	20	0	1.5	1,000	5
May 25, 2022, 3:03:41 PM	10	63.44278	20	0	1.5	1,000	5
May 25, 2022, 3:03:40 PM	10	63.44278	20	0	1.5	1,000	5
May 25, 2022, 3:03:39 PM	10	65.792511	20	0	1.5	1,000	5
May 25, 2022, 3:03:38 PM	8.2	65.792511	20	0	1.5	800	4
May 25, 2022, 3:03:37 PM	6.4	65.453239	20	0	1.5	600	3
May 25, 2022, 3:03:36 PM	6.4	65.453239	20	0	1.5	600	3

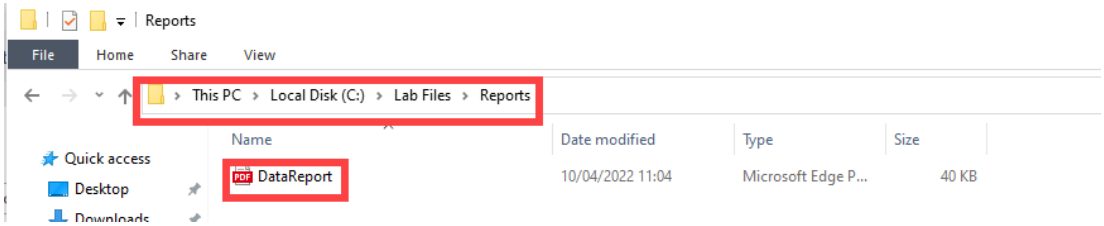
6. The pdf report will open showing the Line logged parameters and the latest alarms. Close the pdf down.

7. Check the **Report Stylesheet2** checkbox.

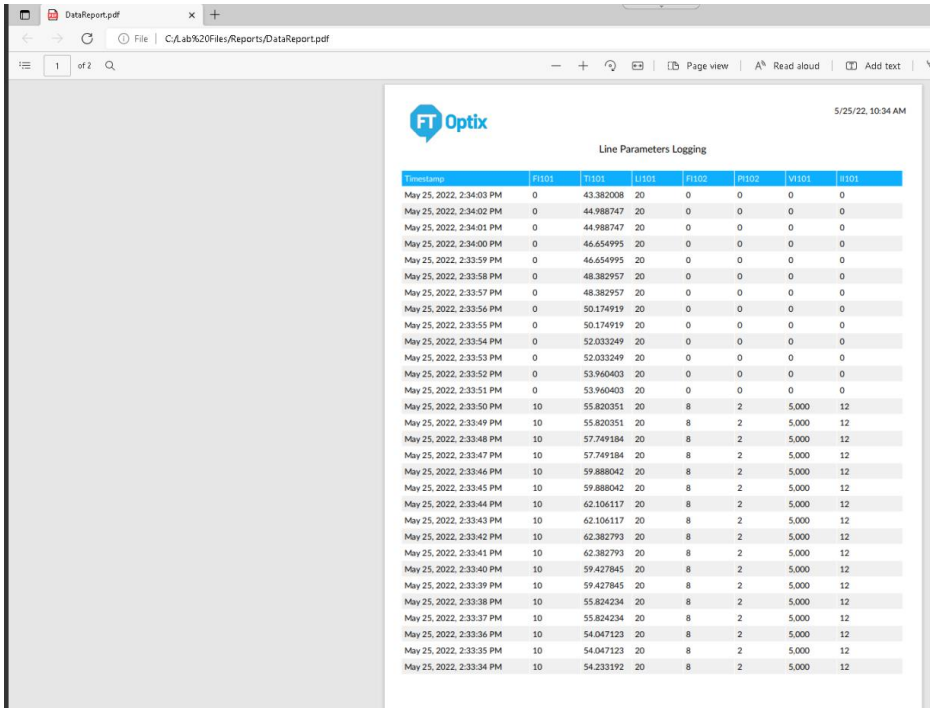


8. Open the folder **C:\Lab Files\Reports**.

   Double click on the **DataReport** pdf.



9. The pdf report will open showing the Line logged parameters and the latest alarms.

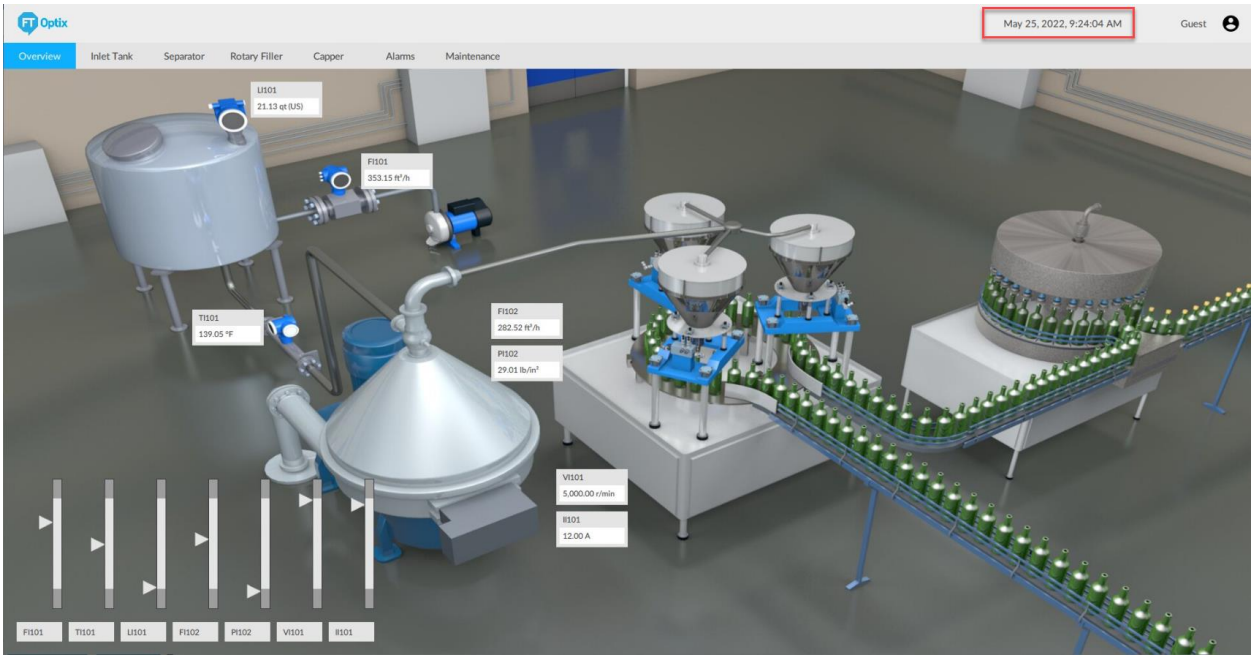


Timestamp	FI101	TI101	LI101	FI102	PI102	VI101	II101
May 25, 2022, 2:34:03 PM	0	43.382008	20	0	0	0	0
May 25, 2022, 2:34:02 PM	0	44.988747	20	0	0	0	0
May 25, 2022, 2:34:01 PM	0	44.988747	20	0	0	0	0
May 25, 2022, 2:34:00 PM	0	46.654995	20	0	0	0	0
May 25, 2022, 2:33:59 PM	0	46.654995	20	0	0	0	0
May 25, 2022, 2:33:58 PM	0	48.382957	20	0	0	0	0
May 25, 2022, 2:33:57 PM	0	48.382957	20	0	0	0	0
May 25, 2022, 2:33:56 PM	0	50.174919	20	0	0	0	0
May 25, 2022, 2:33:55 PM	0	50.174919	20	0	0	0	0
May 25, 2022, 2:33:54 PM	0	52.033249	20	0	0	0	0
May 25, 2022, 2:33:53 PM	0	52.033249	20	0	0	0	0
May 25, 2022, 2:33:52 PM	0	53.960403	20	0	0	0	0
May 25, 2022, 2:33:51 PM	0	53.960403	20	0	0	0	0
May 25, 2022, 2:33:50 PM	10	55.820351	20	8	2	5,000	12
May 25, 2022, 2:33:49 PM	10	55.820351	20	8	2	5,000	12
May 25, 2022, 2:33:48 PM	10	57.749184	20	8	2	5,000	12
May 25, 2022, 2:33:47 PM	10	57.749184	20	8	2	5,000	12
May 25, 2022, 2:33:46 PM	10	59.888042	20	8	2	5,000	12
May 25, 2022, 2:33:45 PM	10	59.888042	20	8	2	5,000	12
May 25, 2022, 2:33:44 PM	10	62.106117	20	8	2	5,000	12
May 25, 2022, 2:33:43 PM	10	62.106117	20	8	2	5,000	12
May 25, 2022, 2:33:42 PM	10	62.382793	20	8	2	5,000	12
May 25, 2022, 2:33:41 PM	10	62.382793	20	8	2	5,000	12
May 25, 2022, 2:33:40 PM	10	59.427845	20	8	2	5,000	12
May 25, 2022, 2:33:39 PM	10	59.427845	20	8	2	5,000	12
May 25, 2022, 2:33:38 PM	10	55.824234	20	8	2	5,000	12
May 25, 2022, 2:33:37 PM	10	55.824234	20	8	2	5,000	12
May 25, 2022, 2:33:36 PM	10	54.047123	20	8	2	5,000	12
May 25, 2022, 2:33:35 PM	10	54.047123	20	8	2	5,000	12
May 25, 2022, 2:33:34 PM	10	54.233192	20	8	2	5,000	12

Scripting - Netlogic

In this section, you will add the Netlogic ***Clock*** from the ***Template Libraries*** to the label ***CurrentTime.*** This label will display the localized date and time.

This will be the result:



Read About NetLogic

Introduction

FactoryTalk Optix™ Studio enables development of customized runtime and design time logics written in C# language. At runtime, using these logics, it is possible to run operations in the information model (for example change the value of a variable when another variable changes status) or outside the information model (for example send an e-mail when a specific variable changes status).

To develop these logics, special objects called NetLogic are used.

What is a NetLogic

A NetLogic is an object that contains C# code to execute at runtime or design time. The following templates are available:

Runtime NetLogic,

To create runtime logics linked to the life cycle of the node that contains it or to define new OPC UA methods.

Note

the OPC UA methods, unlike C# methods, are exposed by NetLogic and can be referenced in any point of the FactoryTalk Optix™ Studio *project at design time. They can then be invoked at runtime according to the project logics or by a OPC UA client.*

Design time NetLogic

to create scripts to execute at design time to automate specific operations (for example read a CSV file containing alarm descriptions to write automatically in the project).

NetLogic and C# classes

A NetLogic contains a C# class with the same name in its code, automatically created by FactoryTalk Optix™ Studio *at design time. When a NetLogic object is renamed,* FactoryTalk Optix™ Studio *also automatically renames the corresponding contained C# class.*

Warning

To change the name of this class, rename the NetLogic in FactoryTalk Optix™ Studio*. Do not change the name of the class in the code as this would compromise its operation.*

For example, a NetLogic called PanelLogic contains the following declaration which defines the PanelLogic class:

```
public class PanelLogic : BaseNetLogic
```

Note

the class in a NetLogic is always derived from the FactoryTalk Optix™ *Platform BaseNetLogic class, i.e. the base class that supplies most of the methods to NetLogic, including the Start and Stop methods*

All C# classes are grouped in a .NET project created and automatically updated by FactoryTalk Optix™ Studio.

NetLogic related to the life cycle of a node

The life cycle of a runtime NetLogic, i.e. its existence at runtime, is equivalent to the life cycle of the node that contains it. Therefore, it exists from the time the parent node is created (for example a Panel object) to the time when the same node is removed. These two moments are represented in the C# code of a NetLogic by the Start() and Stop() methods. In these methods it is therefore possible to define the logics to be executed at runtime when the parent node is created and deleted.

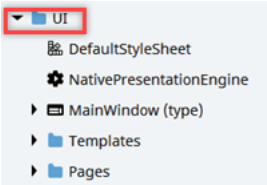
Where to create a NetLogic

NetLogics can be created in any node of the information model. In general, there are two possible positions with different effects at runtime:

In a type (for example in a MotorType). In this case, at runtime the NetLogic exists in each instance of the type until the same instance is deleted.

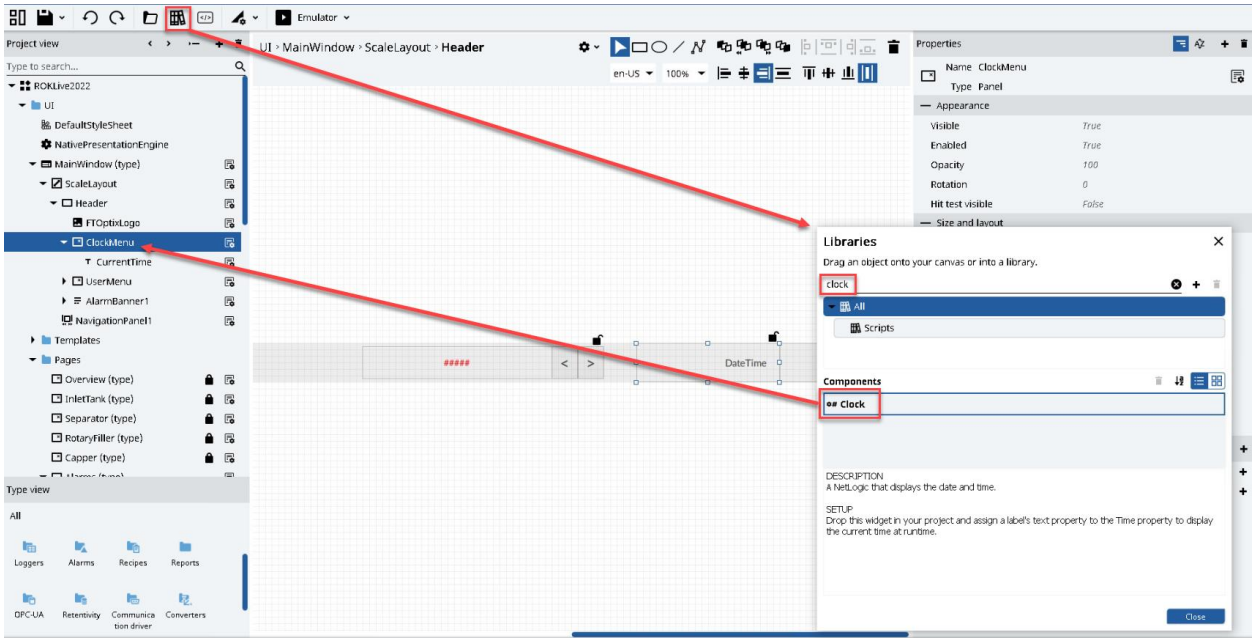
Outside a type (for example, inside a folder, inside other instances, inside the root node of the project). In this case, at runtime the NetLogic is created at project start and deleted at project closure.

1. *Expand* the folder node ***UI***.

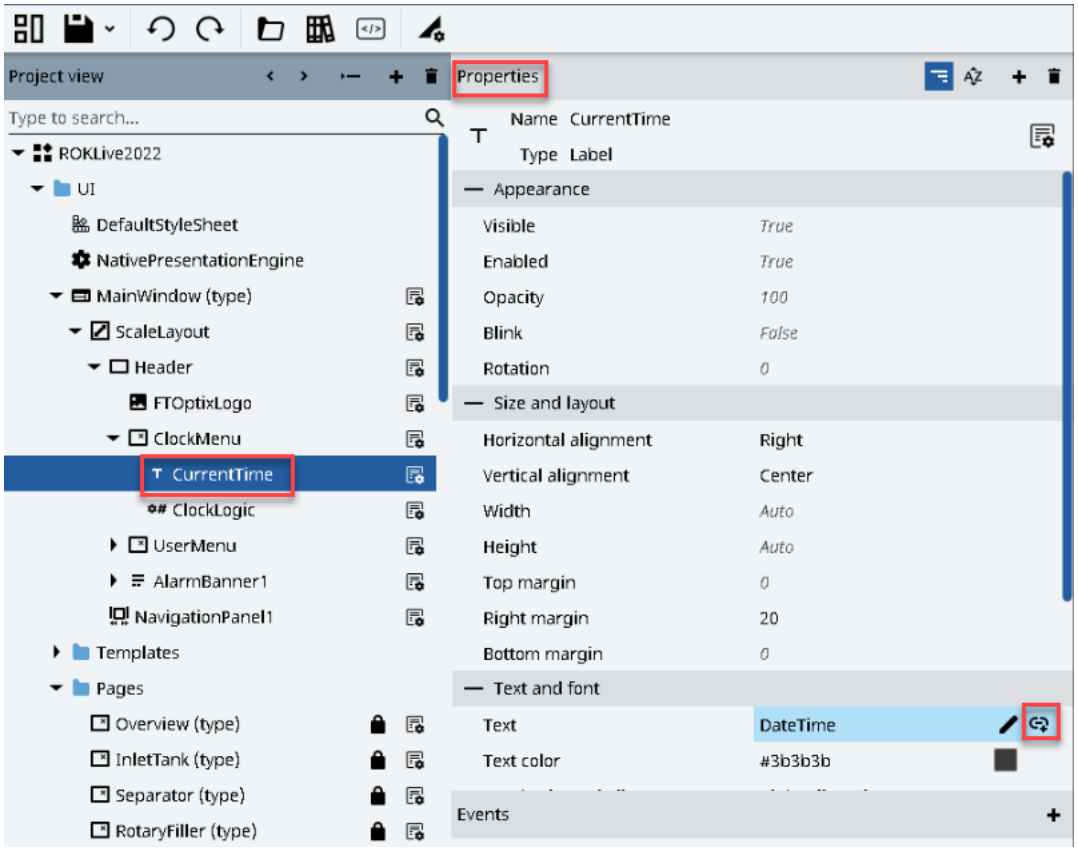


2. *Double Click* on the **Header** under ***MainWindow*** **>> ScaleLayout >> Header.** *Click* on ***ClockMenu*** to show it in the work area.

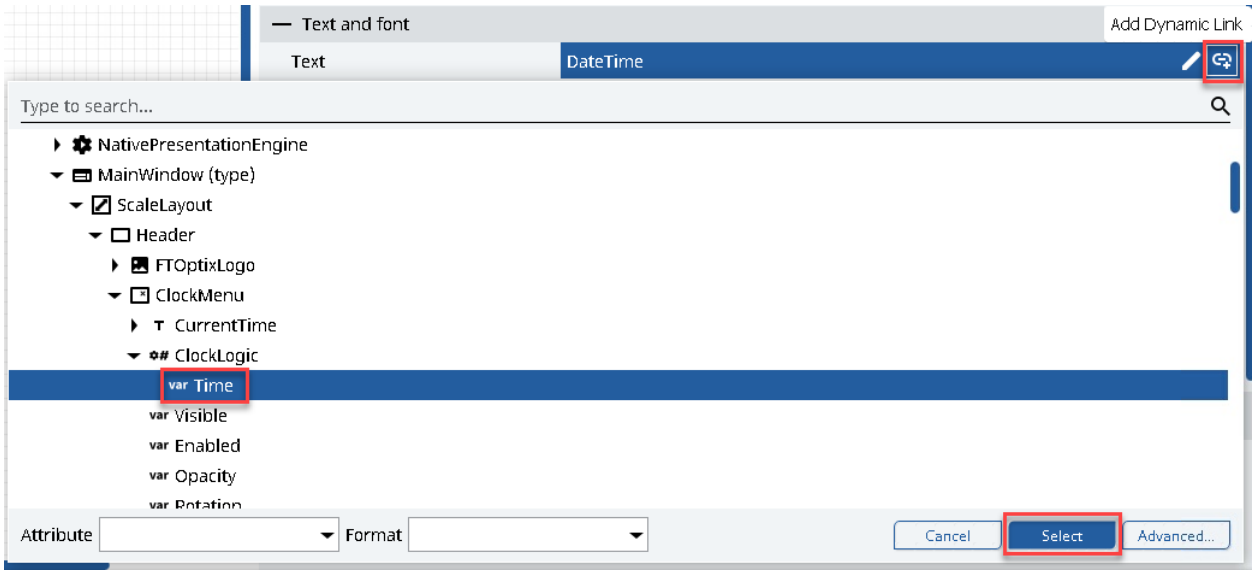
3. *Open* the ***Template Libraries***, type in **clock** in the search bar, drag **Clock** script to the **ClockMenu.**



6. *Select* the label ***CurrentTime*** and go to the property **Text** in the **Properties** panel.



7. *Open* the **Dynamic Link** Popup with the ***chain icon*** (modify) and browse the project tree to the Netlogic ***ClockLogic***. Then *select* the variable **Time**.



8. The configuration of the ***Date and Time*** object is completed.

9. Start your project on the **Emulator** to test your work.

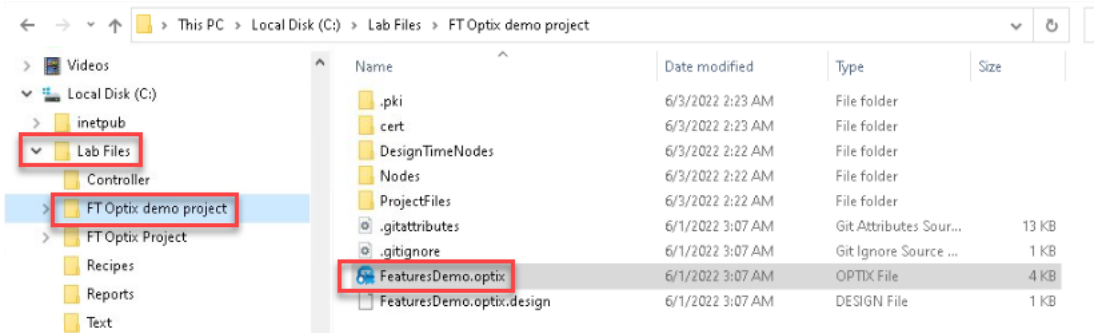
ClockLogic localized Date and Time *– Changing the user logged into the application, the Date and Time format will change accordingly to the User's Locale.*

Template Libraries Netlogics / Widgets *–If you don't know how it works or you don't know how to configure a Netlogic / Widget, remember to look at the corresponding description and setup included in Template Libraries.*

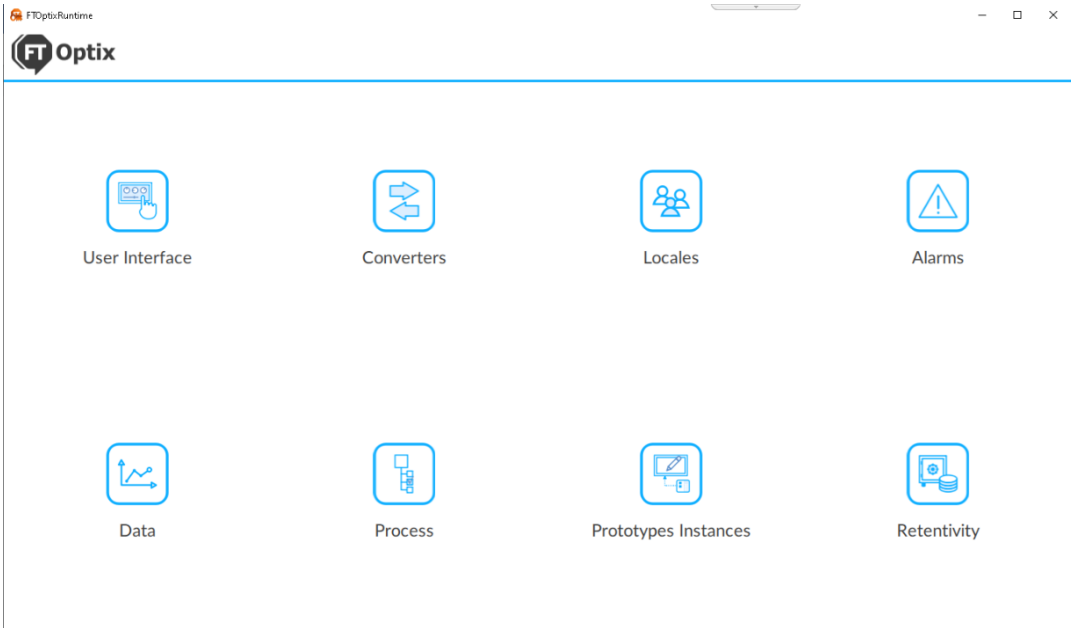
Appendix A - FactoryTalk Optix™ Demo Project

The FactoryTalk Optix™ Demo Project is a project developed to showcase the functionality of FactoryTalk Optix™, this section of the lab is an additional section that allows you to explore this demo.

1. Close down **FactoryTalk Optix™ Studio** and stop the **Emulator** if they are running.
2. Navigate to **C:\Lab Files\FT Optix demo project**. Double Click on the **FT_OptixDemo** project located in the folder.



3. The Demo project will open in FactoryTalk Optix™ Studio.
4. When the project has finished loading, click the **Emulator**.
5. The Project will run in the Emulator, explore the application and if you see some functionality of particular interest then feel free to investigate it in FactoryTalk Optix™ Studio.



Connect with us.

rockwellautomation.com — expanding **human possibility**™

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication XXXX-XX###X-EN-P — Month Year