European Organisation for Astronomical Research in the Southern Hemisphere

Programme: ELT

Project/WP: Cryo HMI Template

ELT Cryo HMI Template User Manual

Document Number: ESO-000000

Document Version: 1

Document Type: Manual (MAN)

Released on: 2024-08-01

Document Classification: ESO Internal [Confidential for Non-ESO Staff]

Owner:	Sagatowski, Jakob
Validated by WPM:	Kiekebusch, Mario
Approved by PM:	
	Name

Release

This document corresponds to CryoHmiTemplate[1] v1.1.0.

Authors

Name	Affiliation
Jakob Sagatowski	ESO

Change Record from previous Version

Affected Section(s)	Changes / Reason / Remarks
TODO	TODO

[1]https://gitlab.eso.org/ifw/ifw-ll

Contents

1 Introduction

Cryo HMI Template is a template project offered to quick start the process of creating an HMI/frontend for a Cryogenic PLC application.

This documents provides instructions on how The TwinCAT template HMI (from now on referred as *TcHmiTemplate* or just *Template*) project works and what it consists of.

1.1 Version history

Table 1: Version history

Date	Notes
2024-06-03	Initial release.
2024-07-10	Added Cryo PLC example project (CryoPlcExampleProject) to replace the PDS PLC project. Fixed bug in UserControls/PumpUserControl.usercontrol, where listener for the state STOPPING was never implemented. Added user control UserControls/SequenceControlFaceplate.usercontrol and associated resources. See chapter for the *sequene faceplates*. Updated dependencies/requirements from projects; The ESO cryo framework 1.0.0 → 1.0.1. The ESO cryo PLC library 0.0.2.10 → 0.0.2.12. Added Pages/EvacuateSequence.content. Added EvacuateSequence page identity to PageIdentity datatype. See chapter *sequence*. Updated Pages/CryogenicController.content by adding an instance of the SequenceControlFaceplate.usercontrol.
2024-09-24	Fixed bug in the UserControls/StirlingCoolerUserControl.usercontrol in where the state symbol was not updated according to the state of the cooler. Added user control UserControls/LakeshoreControllerUserControl.usercontrol. See chapter for the *lakeshore controller user control*. Added faceplate Faceplates/LakeshoreFaceplate.usercontrol. See chapter for the *lakeshore faceplate*. Updated dependencies/requirements from projects; The ESO cryo PLC library 0.0.2.12 → 0.0.2.16. Updated Pages/CryogenicController.content by adding an instance of the LakeshoreControllerUserControl.usercontrol & Faceplates/LakeshoreFaceplate.usercontrol. Changed behaviour of the filtering of event messages, so that when pressing the “Active Alarms” and “All Messages” buttons in the device faceplates also filter based on the instance name of the device and not only on the event class. Added column in the event class to display the source/device name of all events.
2024-10-01	Added user access control for the following: - Service tab faceplate (Engineering on/off). Only engineering group is allowed to turn engineering mode on. Engineering group + operation group can turn engineering mode off. - Sequence control faceplate. The monitoring group is not allowed to press the “Start” + “Stop” buttons. - No Alarm/Warning faceplate. The monitoring group is not allowed to press

1.2 Scope

This document is the user manual for Cryo HMI Template project. The intended audience are ELT users, consortia developers or Cryogenic Engineers.

This Package is one of the deliverables for ESO Cryo Framework, which is composed by:

- **Cryo Library**: PLC Library for standard ESO Cryogenic devices.
- **Cryo HMI Framework Controls**: Collection of HMI Framework controls that will interface with Cryo Library
- **Cryo HMI Template Project**: Start-up project to create an HMI Application. The project comes with a base project structure, UserControls and references to ESO libraries (this document).
- **Cryo PLC Example Project**: PLC project used in conjunction with the Cryo HMI Template Project. There is no hardware (I/O) dependency for this project and as such it can be running locally on the desktop.

1.3 Acronyms

ELT	Extremely Large Telescope
FB	Function Block
GUI	Graphical User Interface
HMI	Human Machine Interface
IFW	Instrument Framework
PLC	Programming Logical Controller

2 Quick Start

The following section shows how to quick start a Cryo HMI Application project based on the template project.

Warning: This manual assumes that the user has already knowledge of TE2000[1]

2.1 Prerequisites

Installed software:

- TwinCAT XAE 3.1.4024.n, where n > 25. The same as IFW.
- TwinCAT 3 HMI Engineering[2] | TE2000: 1.12.760.59.
- The ESO cryo framework (version 1.0.1 or later, NuGet package)
- The ESO cryo PLC library (version 0.0.2.18 or later)
- The ESO cryo PLC HMI gateway library (version 1.0.0.0 or later)

The TcHmiTemplate is a Beckhoff TwinCAT TE2000 project. It was developed using TE2000 version 1.12.760.59. It is highly recommended to use this version to open up the project to minimize compability issues. If this particular version is not available for download from Beckhoff's website, then the local Beckhoff support can provide the correct version.

The ESO Cryo Framework is a NuGet package. Your development environment needs to be able to find it, which it can do through the "TwinCAT HMI Customer" package source, which by default is C:\TwinCAT\Functions\TE2000-HMI-Engineering\References. The ESO cryo framework nuget package (CryoFramework.1.0.1.nupkg) needs to be copied to this folder.

All other NuGet dependencies that the TcHmiTemplate needs are available through the NuGet.org package source. The packages that are used and not pre-installed by default for TE2000 are:

- Beckhoff.TwinCAT.HMI.EventLogger[3]
- JB.TcHmiDynamicPopup[4]

As both packages are public packages available through the NuGet.org[5] package source, they can be automatically installed and no manual installation is necessary.

[1] https://infosys.beckhoff.com/english.php?content=../content/1033/te2000_tc3_hmi_engineering/index.html
[2] https://www.beckhoff.com/en-en/products/automation/twincat/texxxx-twincat-3-engineering/te2000.html
[3] https://www.nuget.org/packages/Beckhoff.TwinCAT.HMI.EventLogger
[4] https://www.nuget.org/packages/JB.TcHmiDynamicPopup
[5] https://www.nuget.org/

2.2 Guidelines

Updates to TcHmiTemplate

It can be safely assumed that there will be changes done to the TcHmiTemplate over the lifespan of the project. One limitation of the TwinCAT HMI (as of now, this will change in a future version of the TwinCAT HMI), is that it's not possible to package the user controls (*.usercontrol) as NuGet packages.

This is different with framework controls as these can be packaged (and versioned) using the conventional NuGet package management, which makes upgrades of the controls easier. This means that any new updates of user controls have to be manually managed.

If the parameters of an user control have been changed (either by new parameters being added, or by existing parameters data definition being updated), the user controls would have to be manually copied over to the users HMI project. If a change in the HMI gateway function block is made and the equivalent user control is changed, it's important to make sure that the mapping is refreshed in the HMI project.

For example, if a data field is added to a gateway HMI function block, it will mean that the ADS definition for that function block will be different than what is currently defined in the project. This will have an impact on the user control, whose parameters will change. As such, if a change of this type is made, it's necessary to refresh the mappings of all HMI gateway function blocks, which is accessible through **TwinCAT HMI** → **Windows** → **TwinCAT HMI Configuration** → **Server Symbols** → **Mapped Symbols**.

Right-click on all the mismatched mappings (visible through a yellow exclamation mark on the symbol) and select Refresh Mapping.

Important: To facilitate upgrades between different versions of the TcHmiTemplate project, ESO will make sure to include release notes between the different releases of the TcHmiTemplate so that it's clear what changes have been made to it.

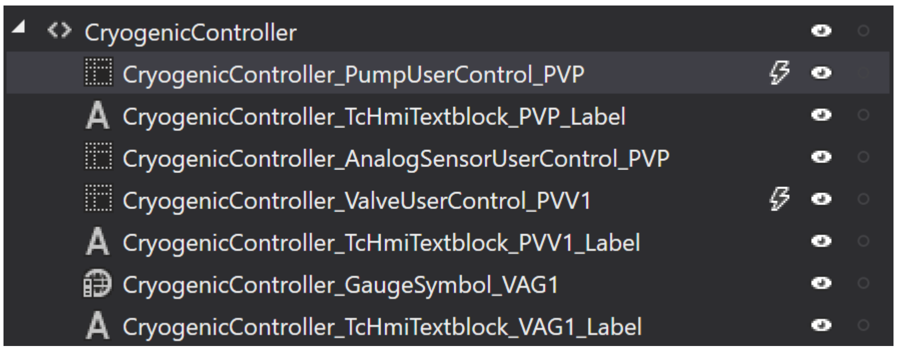
Naming conventions

As every instance of every control's identity is global in the TwinCAT HMI project namespace, it's important to give unique names to every object created in the TwinCAT HMI. The general naming convention for all objects in the TwinCAT HMI is:

```
<ContentName>_<TypeOfControl>_<UniqueName>
```

For example there is one instance of the PumpUserControl called CryogenicController_PumpUserControl_PVP. In other words, there is an instance of the user control PumpUserControl in the content page CryogenicController having the name PVP.

One example of how it could look like using the **document outline**:



CryogenicController
CryogenicController_PumpUserControl_PVP
CryogenicController_TcHmiTextblock_PVP_Label
CryogenicController_AnalogSensorUserControl_PVP
CryogenicController_ValveUserControl_PVV1
CryogenicController_TcHmiTextblock_PVV1_Label
CryogenicController_GaugeSymbol_VAG1
CryogenicController_TcHmiTextblock_VAG1_Label

Fig. 1: Example of several user controls in the CryogenicController content page.

Important: Adhering to these rules will guarantee that every object will not have a name collision with another object (in possibly another completely different content), and it will make sure that every object has a clear and descriptive name (rather than the ambigous auto-generated name by the TwinCAT HMI).

Whitelisting

The TcHmiTemplate project has Whitelisting[6] enabled. With whitelisting, PLC symbols can be explicitly hidden or shown for use in the HMI. This avoids that all the PLC-variables are visible for the TwinCAT HMI, and it also makes it clear in the PLC-code which variables will be displayed in the HMI and which variables will be used to execute a command.

For this to work, the variables that should be whitelisted need to be marked with at least one pragma. There are mainly two pragmas in use in the PLC-code:

```
{attribute 'TcHmiSymbol.Show'}
```

This will tell the HMI that the symbol should be accessible to the HMI. Additionaly to this, the following pragma is also used:

```
{attribute 'TcHmiSymbol.ReadOnly'}
```

This will tell the HMI that this variable can not be written to, and is a ReadOnly variable. This is useful for cases where we want to indicate a value or status for something that we don't want the HMI to change, such as an analog value (such as a temperature). This pragma is not be used for variables that require write access, such as variables to command the system to do something (such as going to manual or automatic mode for a control).

[6] https://infosys.beckhoff.com/english.php?content=../content/1033/te2000_tc3_hmi_engineering/10740009611.html&id=

2.3 Project Setup

Open up the TcHmiTemplate project by opening the file *CryoHmiTemplate.sln* in the project root. If necessary, rename the application as described in the section *Using Template Object*.

Start the PLC application so that the TcHmiTemplate project has the associated variables available to it. The default PLC-runtime settings (ADS) for the HMI project is to connect to the local AmsNetId (127.0.0.1.1.1).

Next, either run the live-view (**TwinCAT HMI** → **Windows** → **TwinCAT HMI Live-View**) or by installing a TwinCAT HMI server (TF2000) and publishing the HMI project (**TwinCAT HMI** → **Publish to TwinCAT HMI Server...**). The latter is recommended as it will give the full functionality, including user management which is important especially for the function of the maintenance mode.

Using template project

The TcHmiTemplate project can be used either as-is or as an inspiration from where the various user controls, JavaScript and examples can be copied from.

If used as-is, it can be renamed by renaming the solution and the TwinCAT HMI project directly in the TcXaeShell. This is however not enough, as there is still one step that needs to be done manually.

For example, let's assume both the solution and the TwinCAT HMI-project is renamed like this:

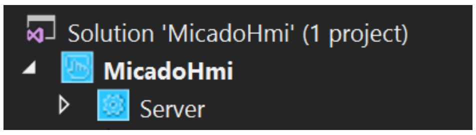


Fig. 2: Result after project is renamed.

In the root of the project however, there will still be a folder called CryoHmiTemplate. It will be necessary to rename this appropriately, for example MicadoHmi.

Now it will be necessary to open the *.sln-file (in this case, the MicadoHmi.sln). In there a line like this will exist:

```
Project("{FE7A1B72-C5B7-4D7C-BB7D-76384D4DE8E1}") = "MicadoHmi", "CryoHmiTemplate\
↪MicadoHmi.hmiproj", "{4CD6F035-8A9F-42E8-9D15-1C7F369338F5}"
```

The CryoHmiTemplate\ will have to be replaced with MicadoHmi\ or whatever the name of the folder is.

If used as inspiration, special care has to be taken to make sure all the data definitions and mappings are also properly created. When linking a variable from the PLC to the HMI, the TwinCAT HMI creates a mapping between the PLC-variable and a local name. Simply copying all objects (such as contents) with their various event listeners will not automatically create these mappings, and they have to be created manually.

3 Template Project

TcHmiTemplate allows for rapid development, it already provides to the user:

1. *HMI Base Application*: An HMI solutions already implemented.
2. *Reusable Elements*: The project comes bundled with a set of UserControls that can be used, copied and/or extended.
3. *Sequence*: Included in the project there are controls to manage sequences.
4. *Content test*: If you need to fit a lot of content into a small screen, this chapter shows how to do it.

3.1 HMI Base Application

TcHmiTemplate consists of the usual elements that every TE2000 project has (Server settings, references, themes etc) and additionaly custom content.

The custom content included is:

- *HMI Layout* default.
- *Navigation Header* that can be extended.
- *Views* for different screen resolutions.
- *Maintenance Mode* configuration.
- *Reusable Elements* as UserControls.

HMI Layout

The standard layout for the HMI application, it is divided into three main parts:

- **Navigation Bar**: Provides the element for navigating the different contents of the system, and additionally a possibility to logout the current user. It's described in more detail at the *Navigation Bar Section*
- **Main Content**: Main display area with the most screen estate, in where the content will change depending on which navigation button is pressed.
- **Status Bar**: Presents the current date/time, the currently logged-in user and the latest created event.

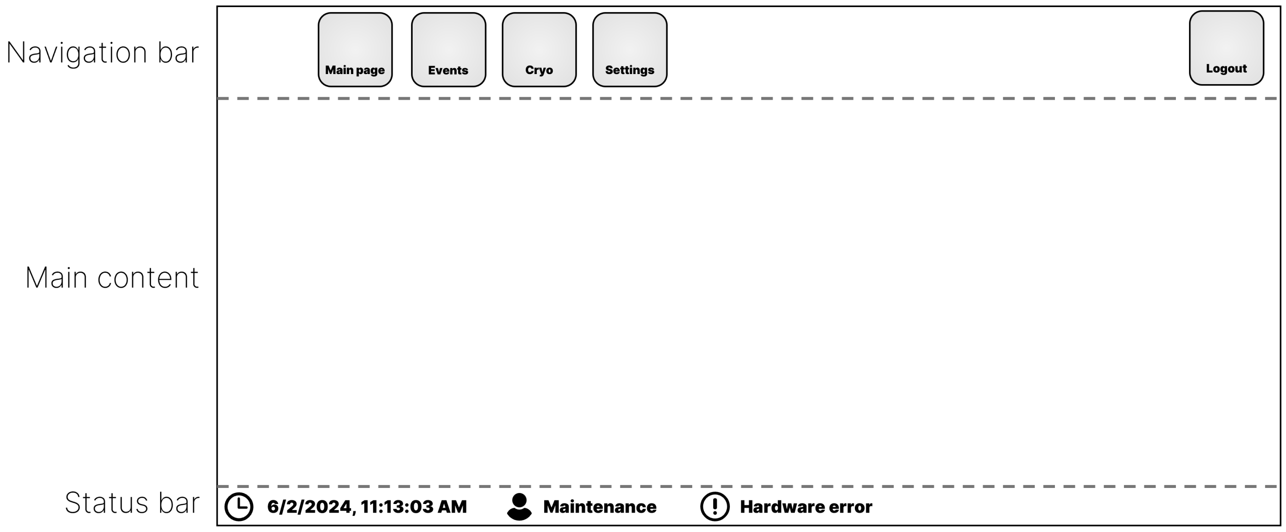


Fig. 1: Main layout concept for the HMI Base Application.

Navigation

The main navigation content is the Pages/Header.content. It consists of several UserControls/NavigationButton.usercontrol. To create a new page for content, first create the page (content) itself. Next add a NavigationButton to the Header. The NavigationButton has three parameters:

1. **Identity**. This uniquely identifies a specific page that should be displayed when that navigation element is selected. It is defined as a data types under Project/PageIdentity (available under **TwinCAT HMI Configuration** → **Data types**). This data type is used in an internal symbol named SelectedPageIdentity in the project.
2. **Label**. This is the text label for the button. Add a text to the localization and use that here.
3. **Icon**. This is the graphical symbol in the navigation element. This is a themed resource (avaiable under **TwinCAT HMI Configuration** → **Themes Resources**), using two different icons, one black and one white, so that it works for both the light and dark theme.

Once the navigation element has been added, it's necessary to add the logic for switching the page when the navigation button has been pressed. This is done in the ControlCabinet.view and ControlRoom.view. In each one of them, there is a event-listener on SelectedPageIdentity that switches the content of the main region to the correct content. Add your page to this event listener and make sure to do this for both views.

Views

It's assumed that the HMI will be running on primarily two different types of operator panels.

1. In the control cabinet
2. In a control room

The first one is assumed to be a small low-resolution display (around 15 inches, 1024x768 resolution) mounted in or close to the electrical cabinet hosting the PLC. The second one would be something in a control room where there is more space, and therefore the display will most likely be larger and thus also the resolution. To have the flexibility to develop for both, the TcHmiTemplate provides two views for this purpose.

1. ControlCabinet.view
2. ControlRoom.view

The ControlCabinet view has been optimized for a resolution of 1024x768, while the ControlRoom view can be used for anything higher than that. Even though the ControlCabinet view is optimized for a resolution of 1024x768, it can be used for higher resolutions as well. In this case, the HMI will simply be scaled to fit the screen. The disadvantage of this is that it's not an optimal way to use the screen estate on a high resolution display.

Using the ControlRoom view gives the option to maximize the usage of the available screen estate which can be shown in this example.

As can be seen, the ControlRoom view has much more space for the navigation elements (at the top), the main content (in this case the events) and more space for the status bar (at the bottom).

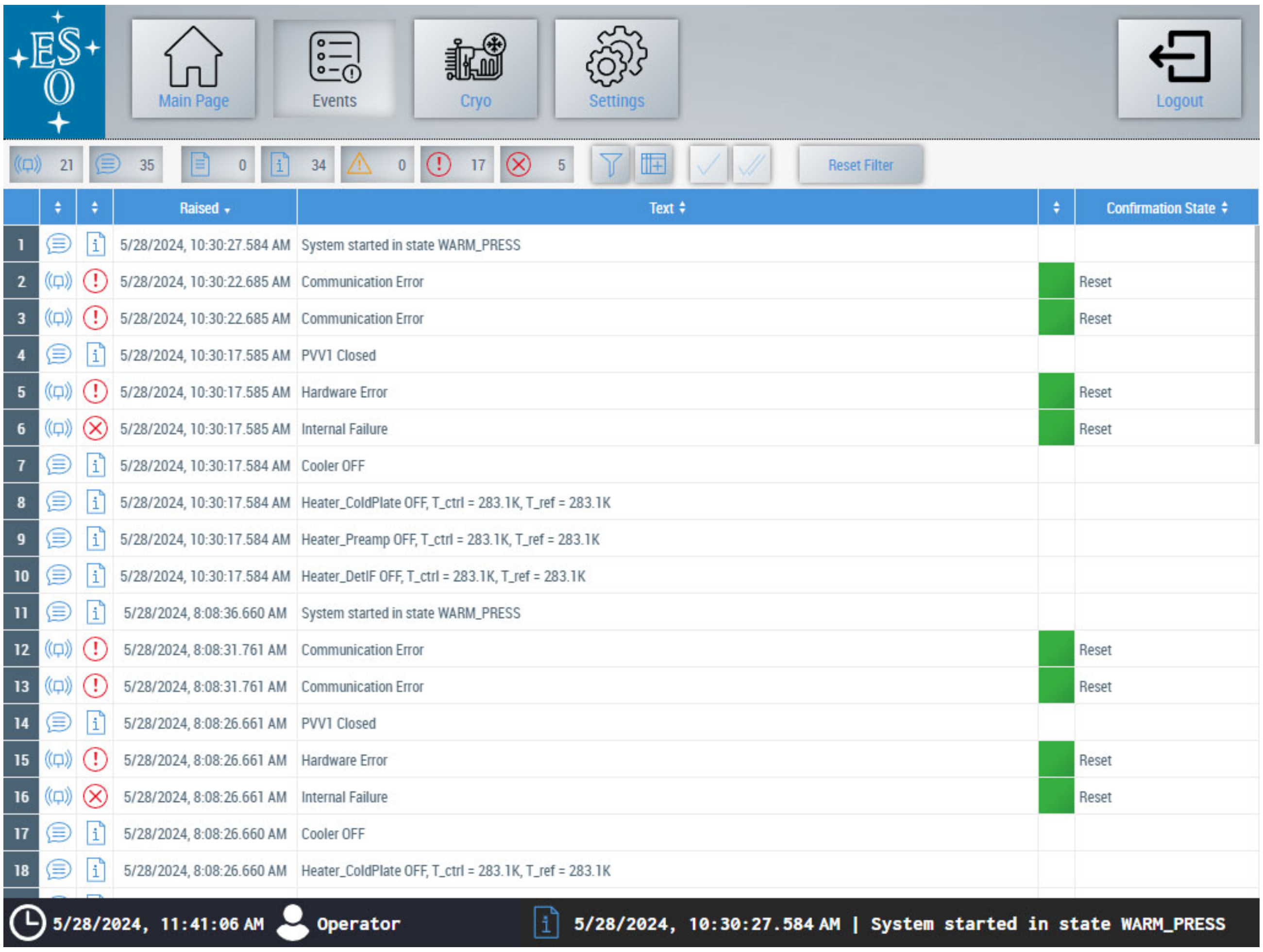
The different views can be accessed from the client browser through adding ?View=<ViewName> to the URL, in other words like this:

- https://127.0.0.1:1020/?View=ControlCabinet
- https://127.0.0.1:1020/?View=ControlRoom

The advantage of this is that for the commisioned system it's possible to make a shortcut to the web browser that will autostart during boot of that system and include the correct view for that system. So for example, assuming we would use Chrome (or better yet, the ungoogled chromium browser) and we wanted to create a shortcut that autostarted when TwinCAT boots at full screen we could achieve this by creating this shortcut:

```
chrome.exe --kiosk https://localhost:1020?View=ControlCabinet --incognito --disable-pinch --
↪overscroll-historynavigation=0
```

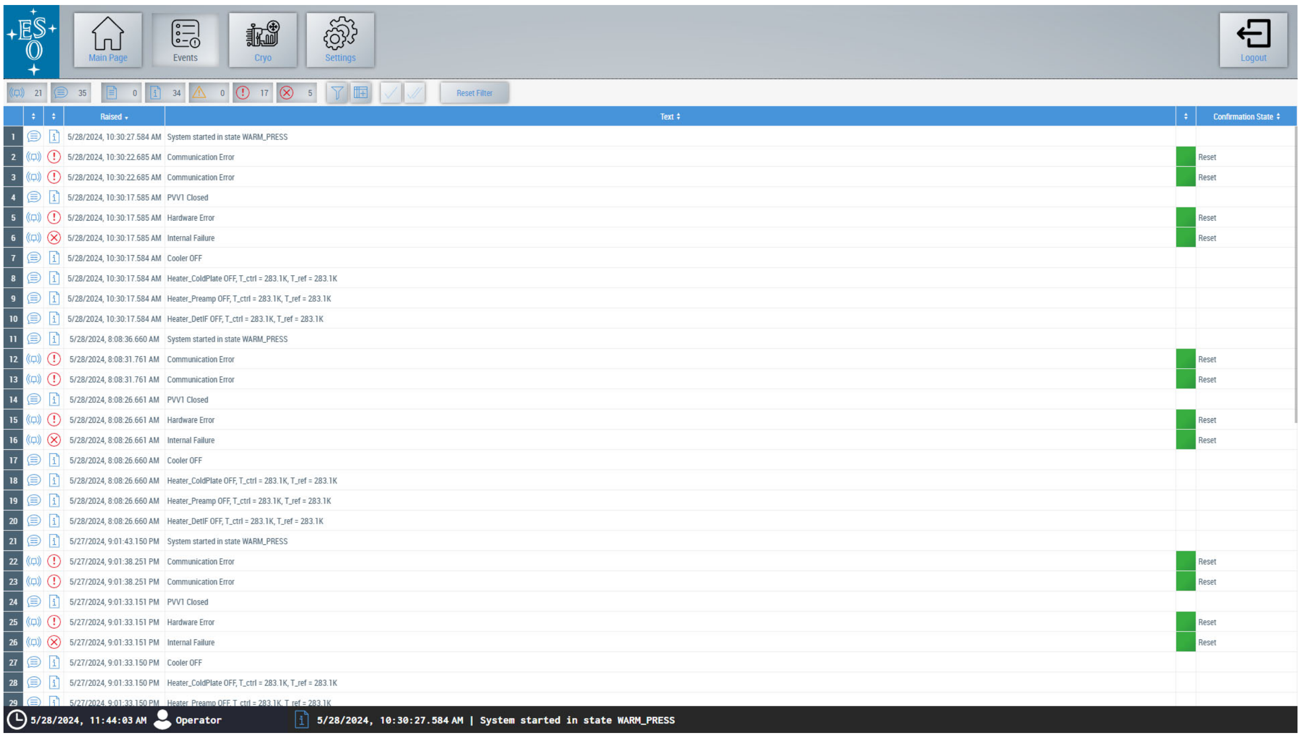
This would fire up Chromium in so-called kiosk-mode (full-screen, not possible to minimize or close) and using the ControlCabinet view. If we wanted to create another shortcut for other system (i.e. for the control room of the telescope) we would simply replace the ControlCabinet with ControlRoom.



			Raised	Text		Confirmation State
1	💬	ℹ	5/28/2024, 10:30:27.584 AM	System started in state WARM_PRESS		
2	((□))	❗	5/28/2024, 10:30:22.685 AM	Communication Error	🟩	Reset
3	((□))	❗	5/28/2024, 10:30:22.685 AM	Communication Error	🟩	Reset
4	💬	ℹ	5/28/2024, 10:30:17.585 AM	PVV1 Closed		
5	((□))	❗	5/28/2024, 10:30:17.585 AM	Hardware Error	🟩	Reset
6	((□))	⊗	5/28/2024, 10:30:17.585 AM	Internal Failure	🟩	Reset
7	💬	ℹ	5/28/2024, 10:30:17.584 AM	Cooler OFF		
8	💬	ℹ	5/28/2024, 10:30:17.584 AM	Heater_ColdPlate OFF, T_ctrl = 283.1K, T_ref = 283.1K		
9	💬	ℹ	5/28/2024, 10:30:17.584 AM	Heater_Preamp OFF, T_ctrl = 283.1K, T_ref = 283.1K		
10	💬	ℹ	5/28/2024, 10:30:17.584 AM	Heater_DetIF OFF, T_ctrl = 283.1K, T_ref = 283.1K		
11	💬	ℹ	5/28/2024, 8:08:36.660 AM	System started in state WARM_PRESS		
12	((□))	❗	5/28/2024, 8:08:31.761 AM	Communication Error	🟩	Reset
13	((□))	❗	5/28/2024, 8:08:31.761 AM	Communication Error	🟩	Reset
14	💬	ℹ	5/28/2024, 8:08:26.661 AM	PVV1 Closed		
15	((□))	❗	5/28/2024, 8:08:26.661 AM	Hardware Error	🟩	Reset
16	((□))	⊗	5/28/2024, 8:08:26.661 AM	Internal Failure	🟩	Reset
17	💬	ℹ	5/28/2024, 8:08:26.660 AM	Cooler OFF		
18	💬	ℹ	5/28/2024, 8:08:26.660 AM	Heater_ColdPlate OFF, T_ctrl = 283.1K, T_ref = 283.1K		

```
5/28/2024, 11:41:06 AM  Operator        5/28/2024, 10:30:27.584 AM | System started in state WARM_PRESS
```

Fig. 2: **ControlCabinet View**, showing the system events



			Raised	Text		Confirmation State
1	💬	ⓘ	5/28/2024, 10:30:27.584 AM	System started in state WARM_PRESS		
2	📢	❗	5/28/2024, 10:30:22.685 AM	Communication Error	🟩	Reset
3	📢	❗	5/28/2024, 10:30:22.685 AM	Communication Error	🟩	Reset
4	💬	ⓘ	5/28/2024, 10:30:17.585 AM	PVV1 Closed		
5	📢	❗	5/28/2024, 10:30:17.585 AM	Hardware Error	🟩	Reset
6	📢	⊗	5/28/2024, 10:30:17.585 AM	Internal Failure	🟩	Reset
7	💬	ⓘ	5/28/2024, 10:30:17.584 AM	Cooler OFF		
8	💬	ⓘ	5/28/2024, 10:30:17.584 AM	Heater_ColdPlate OFF, T_ctrl = 283.1K, T_ref = 283.1K		
9	💬	ⓘ	5/28/2024, 10:30:17.584 AM	Heater_Preamp OFF, T_ctrl = 283.1K, T_ref = 283.1K		
10	💬	ⓘ	5/28/2024, 10:30:17.584 AM	Heater_DetIF OFF, T_ctrl = 283.1K, T_ref = 283.1K		
11	💬	ⓘ	5/28/2024, 8:08:36.660 AM	System started in state WARM_PRESS		
12	📢	❗	5/28/2024, 8:08:31.761 AM	Communication Error	🟩	Reset
13	📢	❗	5/28/2024, 8:08:31.761 AM	Communication Error	🟩	Reset
14	💬	ⓘ	5/28/2024, 8:08:26.661 AM	PVV1 Closed		
15	📢	❗	5/28/2024, 8:08:26.661 AM	Hardware Error	🟩	Reset
16	📢	⊗	5/28/2024, 8:08:26.661 AM	Internal Failure	🟩	Reset
17	💬	ⓘ	5/28/2024, 8:08:26.660 AM	Cooler OFF		
18	💬	ⓘ	5/28/2024, 8:08:26.660 AM	Heater_ColdPlate OFF, T_ctrl = 283.1K, T_ref = 283.1K		
19	💬	ⓘ	5/28/2024, 8:08:26.660 AM	Heater_Preamp OFF, T_ctrl = 283.1K, T_ref = 283.1K		
20	💬	ⓘ	5/28/2024, 8:08:26.660 AM	Heater_DetIF OFF, T_ctrl = 283.1K, T_ref = 283.1K		
21	💬	ⓘ	5/27/2024, 9:01:43.150 PM	System started in state WARM_PRESS		
22	📢	❗	5/27/2024, 9:01:38.251 PM	Communication Error	🟩	Reset
23	📢	❗	5/27/2024, 9:01:38.251 PM	Communication Error	🟩	Reset
24	💬	ⓘ	5/27/2024, 9:01:33.151 PM	PVV1 Closed		
25	📢	❗	5/27/2024, 9:01:33.151 PM	Hardware Error	🟩	Reset
26	📢	⊗	5/27/2024, 9:01:33.151 PM	Internal Failure	🟩	Reset
27	💬	ⓘ	5/27/2024, 9:01:33.150 PM	Cooler OFF		
28	💬	ⓘ	5/27/2024, 9:01:33.150 PM	Heater_ColdPlate OFF, T_ctrl = 283.1K, T_ref = 283.1K		
29	💬	ⓘ	5/27/2024, 9:01:33.150 PM	Heater_Preamp OFF, T_ctrl = 283.1K, T_ref = 283.1K		

```
5/28/2024, 11:44:03 AM   Operator   5/28/2024, 10:30:27.584 AM | System started in state WARM_PRESS
```

Fig. 3: **ControlRoom View**, showing the system events

Maintenance mode

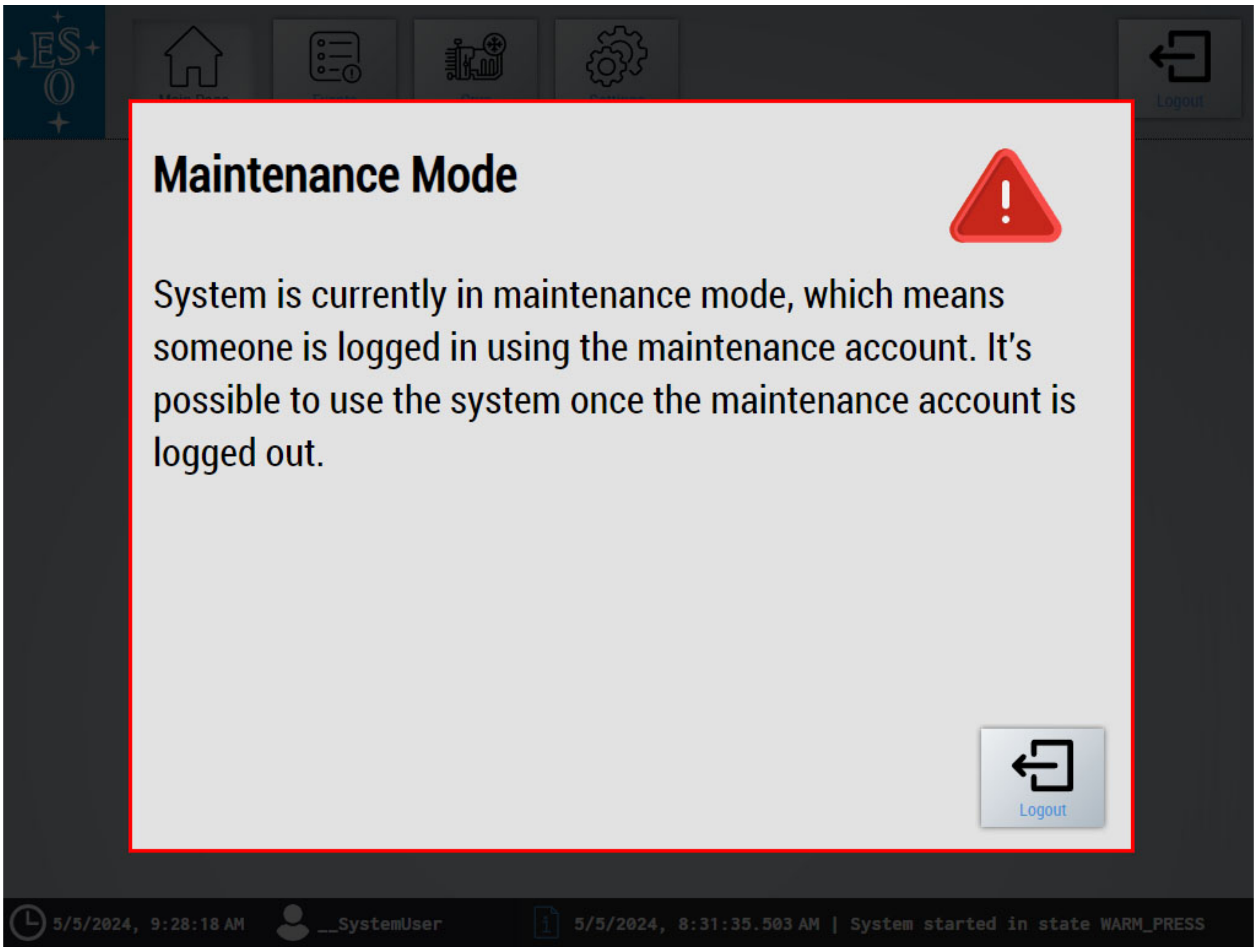
If an operator at the instrument logs in into the system, we want to be sure that no-one else can remotely control the instrument at the same time. As the TwinCAT HMI is based on web technology, it is technically possible to have both a local operator and remote operator logged in into and controlling the system at the same time. To solve this problem, the concept of a maintenance user was created. The maintenance user should only be used at the instrument platform.

The idea is simple; when a maintenance user logs in, everyone else that is not a maintenance user will get a popup that fills the whole screen:

This will cover the whole screen and it's not possible to click or interact with any other elements in the HMI. The only thing that any other user can do is to click/press on the Logout button or wait for the Maintenance user to **logout** (which if it happens, would make the popup go away automatically).

This functionality is implemented in the following way:

1. An user with name **"Maintenance"** was added to the Users in the TwinCAT HMI configuration.
2. A boolean variable was added in the PLC program $\mathrm{MAIN.bMaintenanceUserLoggedIn}$.
3. A global event in the TwinCAT HMI was added that fires of when the HMI is initialized. This consists of a small JavaScript that sets the PLC-variable to **TRUE** if the user that has logged in is the Maintenance user.



Maintenance Mode

System is currently in maintenance mode, which means someone is logged in using the maintenance account. It's possible to use the system once the maintenance account is logged out.

```
5/5/2024, 9:28:18 AM   __SystemUser   5/5/2024, 8:31:35.503 AM | System started in state WARM_PRESS
```

Fig. 4: When in **Maintenance Mode** any other user gets this popup.

4. A piece of JavaScript is added on the .onPressed-event of the **Logout** button that sets the PLC-variable to FALSE if the user that logs out is the Maintenance user.
5. An user control InMaintenanceModePopupUserControl.usercontrol was created which is the popup that shows up on every user that is not maintenance (if maintenance user is logged in).
6. An instance of this user control is created in both the ControlCabinet and ControlRoom views, and an event listener is added on them to show the popup if the PLC-variable is set to true (but only if the user is not the maintenance user).

3.2 Reusable Elements

TcHmiTemplate provides a set of content made to be used as TcHmi UserControls[7]. They are developed with direct compatibility with the PLC devices. The user controls consist of two categories.

1. *Controls*
2. *Faceplates*

Controls

The controls are wrappers around some of the controls of the CryoHmiFramework (NuGet package). They're implemented as user controls (.usercontrol) in the HMI Project. They provide access to the functionality of the framework controls by directly linking them to a PLC variable using the HMI gateway function blocks, which are part of the **cryohmi library**. When the user control is linked to a PLC-variable, it gets all the information it needs to display correctly.

For example, if an instance of the *PumpUserControl* is created in the HMI, with the unique identity CryogenicController_PumpUserControl_PVP, it's necessary to link it a PLC-variable on the PLC. In the example below, an instance of the FB_HMI_CRYO_PUMP (available in the cryohmi library) named hmi_pump_pvp has been created in the path MAIN.pds_cryo_ctrl.

The parameter PumpHmiObject of the PumpUserControl is to be linked to the PLC-variable (through the usual mapping in TwinCAT HMI).

On top of requiring an PLC-variable to be set as parameter, every user control has a few other parameters that can be changed which are related to how the user control should be shown. This could for example be direction and stroke thickness of the symbol.

To use a control, simply drag and drop one of the user controls (for example PumpUserControl or ValveUserControl) into a content and set the parameters for it.

[7] https://infosys.beckhoff.com/english.php?content=../content/1033/te2000_tc3_hmi_engineering/3477219979.html

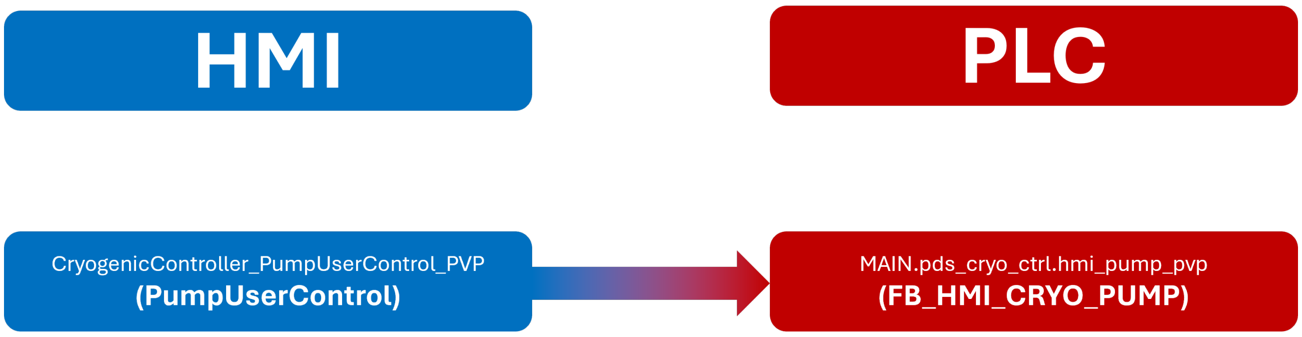

HMI
PLC
CryogenicController_PumpUserControl_PVP
(PumpUserControl)
MAIN.pds_cryo_ctrl.hmi_pump_pvp
(FB_HMI_CRYO_PUMP)

Fig. 5: Representation of PumpUserControl instance CryogenicController_PumpUserControl_PVP binding with the PLC device of type FB_HMI_CRYO_PUMP.

PumpUserControl

Pump user control with all interfaces needed to be connected to an instance of the FB_HMI_CRYO_PUMP function block in the HMI gateway library.

Table 1: PumpUserControl Interface Table

Parameter	Description
PumpHmiObject	To be linked to an instance of the FB_HMI_CRYO_PUMP function block in the PLC project.
SymbolType	See table “Pump interface Table” in document “Cryo HMI Framework Controls”.
Direction	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
SymbolMargin-Size	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
StrokeThickness	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.

StirlingCoolerUserControl

Stirling cooler user control with all interfaces needed to be connected to an instance of the FB_HMI_CRYO_COOLER function block in the HMI gateway library.

Table 2: StirlingCoolerUserControl Interface Table

Parameter	Description
StirlingCoolerHmiObject	To be linked to an instance of the FB_HMI_CRYO_COOLER function block in the PLC project.
Direction	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
SymbolMarginSize	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
StrokeThickness	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.

ValveUserControl

Valve user control with all interfaces needed to be connected to an instance of the FB_HMI_CRYO_VALVE function block in the HMI gateway library.

Table 3: ValveUserControl Interface Table

Parameter	Description
ValveHmiObject	To be linked to an instance of the FB_HMI_CRYO_VALVE function block in the PLC project.
SymbolType	See table “Valve interface Table” in document “Cryo HMI Framework Controls”.
Direction	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
SymbolMarginSize	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
StrokeThickness	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
ShowFeedback	See table “Valve interface Table” in document “Cryo HMI Framework Controls”

LakeshoreControllerUserControl

Lakeshore user control with all interfaces needed to be connected to an instance of the FB_HMI_CRYO_LAKESHORE function block in the HMI gateway library.

Table 4: LakeshoreController Interface Table

Parameter	Description
LakeshoreControllerHmiObject	To be linked to an instance of the FB_HMI_CRYO_LAKESHORE function block in the PLC project.
Label	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
SymbolMarginSize	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.
StrokeThickness	See table “Common Controls Interface Table” in document “Cryo HMI Framework Controls”.

AnalogSensorUserControl

This is a wrapper for the Analog Sensor framework control.

Table 5: AnalogSensorUserControl Interface Table

Parameter	Description
Value	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
DataValid	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Simulated	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Notation	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Decimals	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Signed	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Unit	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
UnitTextboxWidth	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.

HeaterUserControl

This is a wrapper for the Heater framework control.

Table 6: HeaterUserControl Interface Table

Parameter	Description
HeaterHmiObject	To be linked to an instance of the FB_HMI_CRYO_HEATER function block in the PLC project.
Value	See table "Heater interface Table" in document "Cryo HMI Framework Controls".
Direction	See table "Common Controls Interface Table" in document "Cryo HMI Framework Controls".
SymbolMargin-Size	See table "Common Controls Interface Table" in document "Cryo HMI Framework Controls".
StrokeThickness	See table "Common Controls Interface Table" in document "Cryo HMI Framework Controls".
TextFontSize	See table "Heater interface Table" in document "Cryo HMI Framework Controls".

PdsHeaterUserControl

This is a wrapper for the Heater framework control, specifically done for the PDS application, which the PLC example project is based on.

Table 7: PdsHeaterUserControl Interface Table

Parameter	Description
HeaterHmiObject	To be linked to an instance of the FB_HMI_PDS_CRYO_HEATER function block in the PLC project.
Value	See table "Heater interface Table" in document "Cryo HMI Framework Controls".
Direction	See table "Common Controls Interface Table" in document "Cryo HMI Framework Controls".
SymbolMargin-Size	See table "Common Controls Interface Table" in document "Cryo HMI Framework Controls".
StrokeThickness	See table "Common Controls Interface Table" in document "Cryo HMI Framework Controls".
TextFontSize	See table "Heater interface Table" in document "Cryo HMI Framework Controls".

PressureUserControl

This is a wrapper for the Analog Sensor framework control specifically targeted for pressure values.

Table 8: PressureUserControl Interface Table

Parameter	Description
PressureHmiObject	To be linked to an instance of the FB_HMI_CRYO_PRESSURE_VOTE function block in the PLC project.
Notation	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Decimals	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Signed	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
UnitTextboxWidth	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.

TemperatureUserControl

This is a wrapper for the Analog Sensor framework control specifically targeted for temperature values.

Table 9: TemperatureUserControl Interface Table

Parameter	Description
PressureHmiObject	To be linked to an instance of the FB_HMI_CRYO_TEMPERATURE function block in the PLC project.
Notation	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Decimals	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
Signed	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.
UnitTextboxWidth	See table “Analog Sensor Control Interface Table” in document “Cryo HMI Framework Controls”.

Faceplates

Faceplates are user controls to show certain information related to a function. There are two kinds of faceplates:

1. *Control faceplates*
2. *Sequence faceplates*

Control faceplates

The control faceplates are small popups that are shown when the user presses a control (such as a pump or a valve). They provide additional information about a certain control and additionaly provide some control functionality (such as ON/OFF, Engineering Mode ON/OFF). They're implemented as user controls ($.\mathrm{usercontrol}$) in the TwinCAT HMI.

The basic layout of a faceplate looks as follows:

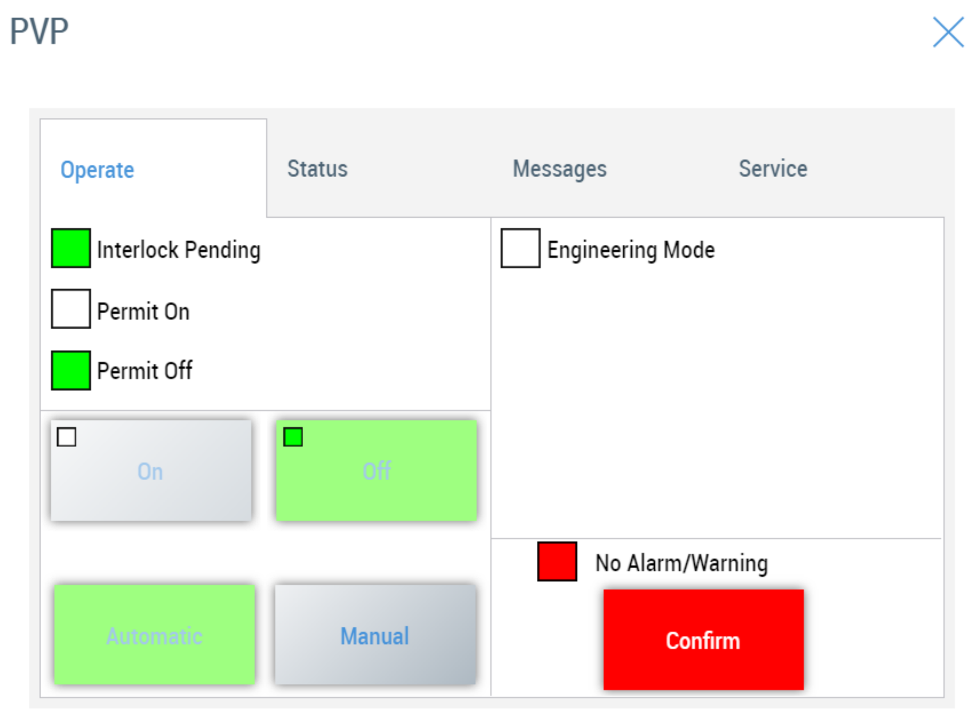


Fig. 6: Faceplate example of a device named PVP. Currently on Tab **Operate**

Most faceplates have three tabs:

1. Operate
2. Messages

3. Service

Additionally, some faceplates have a Status tab, to indicate device status.

The **Operate Tab** is used to see the overall status of the device and a possibility to operate it. From here it's possible to control the device manually and see the current actuation status. It's also possible to switch the mode between automatic and manual for the device. The small square in the upper left corner of the On and Off indicate what the setpoint for the automatic control is. In the screenshot above, the automatic mode wants the pump to be off as the small square inside the Off button is green. The Confirm button is used to confirm all alarms and warnings. Note that this will only confirm the associated events in the event logger, and not clear them. Clearing can only be done automatically (given that the conditions for clearing them are met).

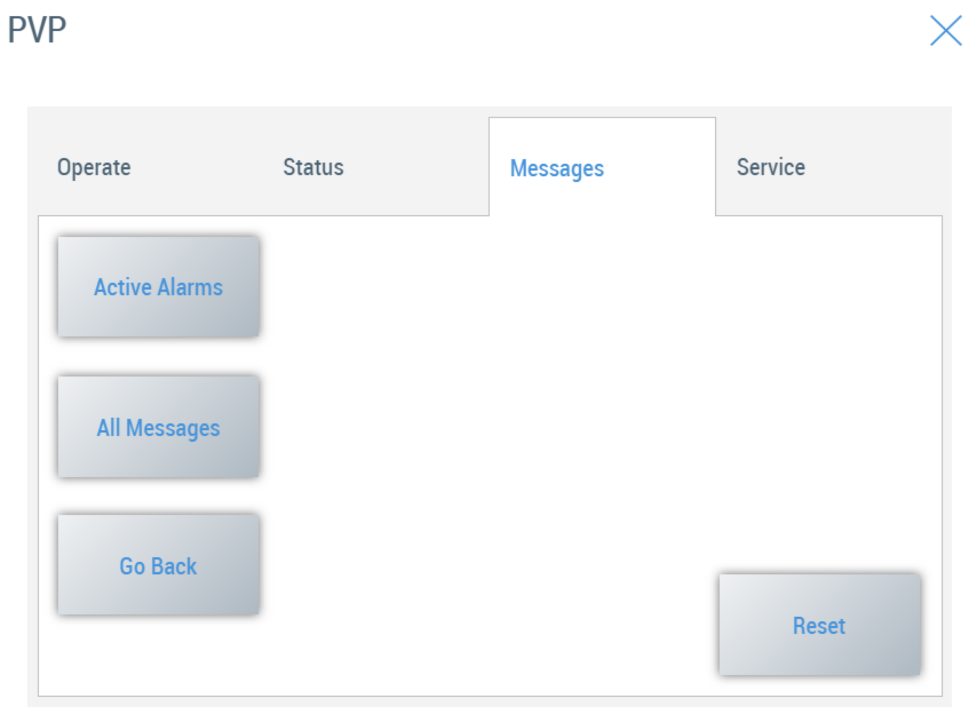


Fig. 7: Faceplate example of a device named PVP. Currently on Tab **Messages**

The **Messages Tab** is used to control what events for the device associated with the faceplate should be displayed in the event tab. By clicking on the Active Alarms button, the event page will be shown and only all events associated to that device that are not cleared (and thus active) will be displayed. Clicking the All Messages button will show all messages related to this device, including the ones that are cleared. Clicking the Reset button will show all events in the event logger, in other words it will display all events and not just the ones related to this device. Clicking the Go Back button will return you to the previously selected page, which is useful if you want to be able to switch quickly between the event page and the page of where the faceplate was created.

The **Service Tab** is used to enter and exit engineering mode. Note that it's only possible to enter engineering mode if the device is in manual mode. Upon entering engineering mode, an orange

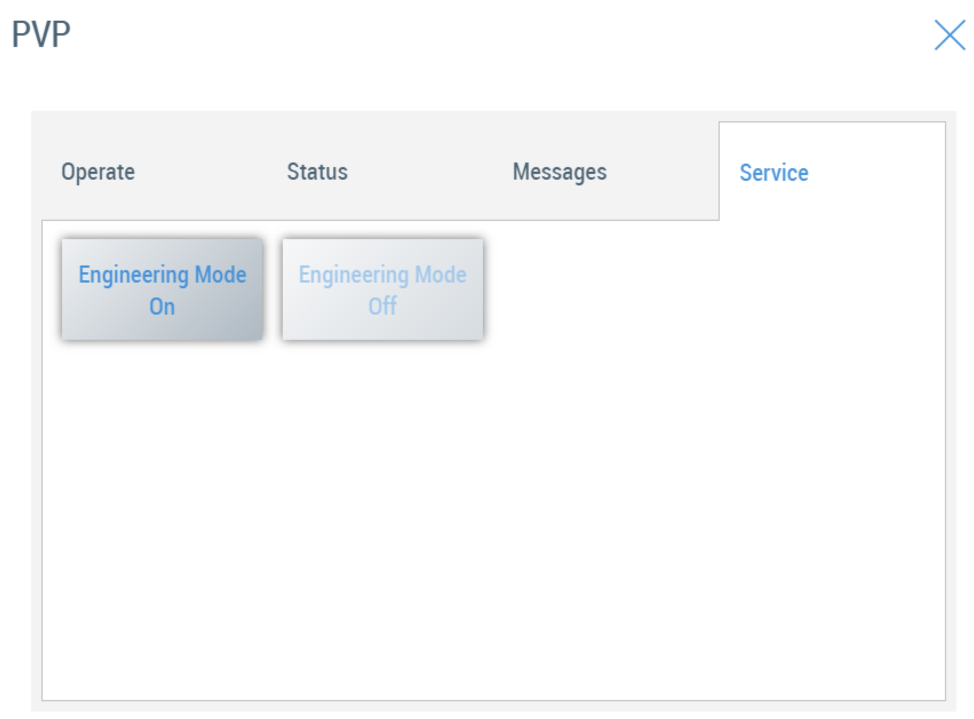


Fig. 8: Faceplate example of a device named PVP. Currently on Tab **Service**

frame will indicate that the device is in engineering mode.

Note that this orange frame is not only visible in this tab but across the whole faceplate. The Lakeshore controller faceplate does not have the service tab as it's not possible to control the device.

The **Status Tab** is used to show device status. This is not available for all device faceplates. This example is shown for the pump faceplate.

In the very same way as for the controls, a faceplate requires an equivalent function block in the HMI to be linked to. For every user control that can be interacted with (such as pump, valve, cooler, . . .) there is an equivalent faceplate user control.

To use a faceplate user control, it needs a context in where the contents of the user control should be shown. This is achieved by displaying it in a popup, which is realized using the JB.TcHmiDynamicPopup[8]. The dynamic popup is instantiated when the user clicks on the user control, more precisely the $.\mathrm{onPressed}()$-event. Here it's necessary to create an instance of the DynamicPopup by drag-and-dropping the function $\mathrm{CreatePopup}()$ from the DynamicPopup category amongst the JavaScript functions.

What fields/parameters that are necessary to enter to create a popup is documented in the official documentation[9] for the TcHmiDynamicPopup, however we will clarify some things for the $\mathrm{targetFile}$-

[8] https://www.nuget.org/packages/JB.TcHmiDynamicPopup
[9] https://www.nuget.org/packages/JB.TcHmiDynamicPopup

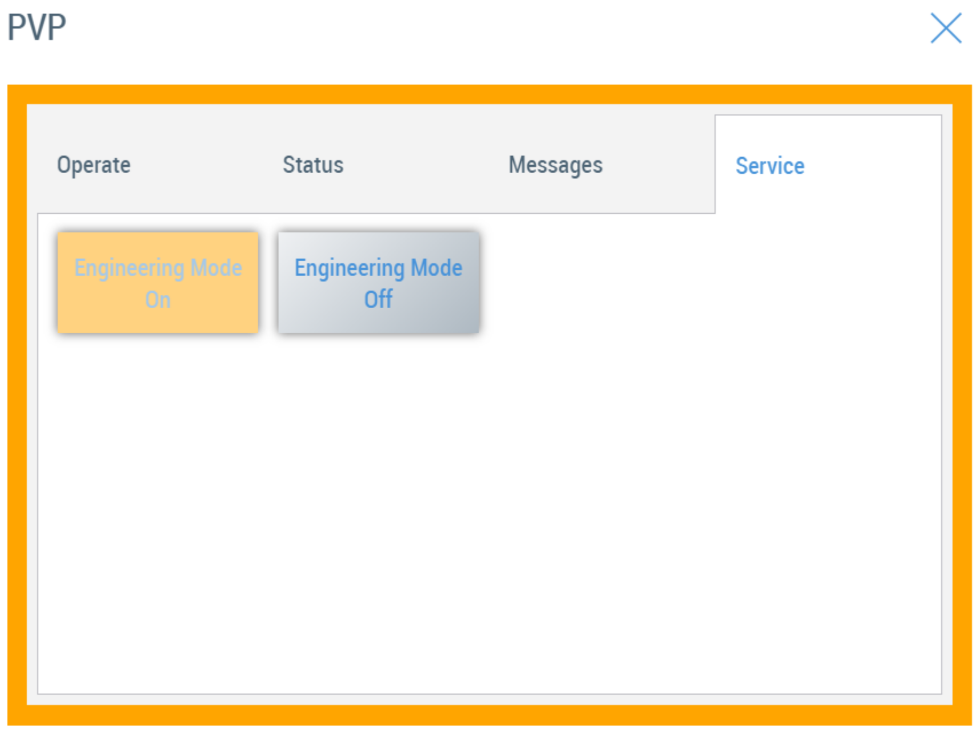


Fig. 9: Faceplate example of a device named PVP. The orange colored border signals that Engineering Mode is activated

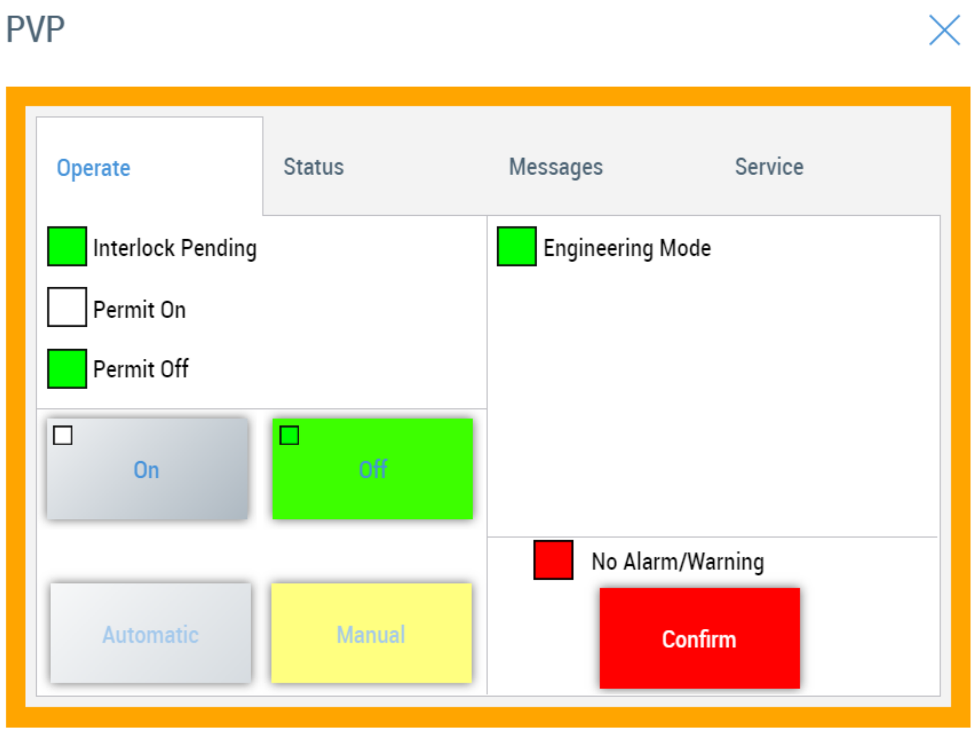
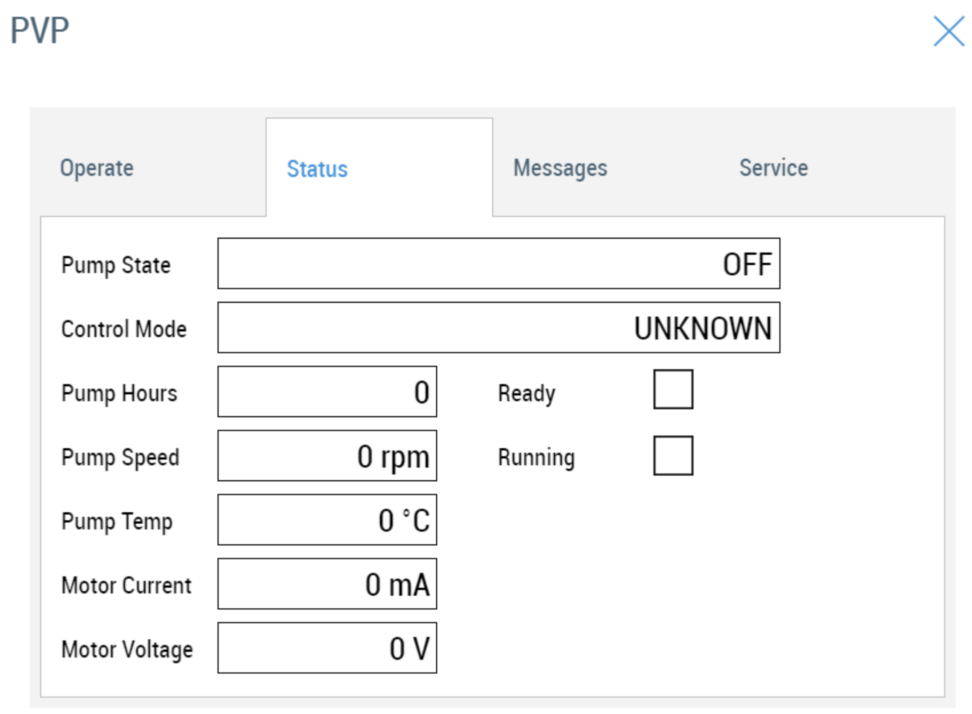


Fig. 10: Faceplate example of a device named PVP. The orange colored border signals that Engineering Mode is activated

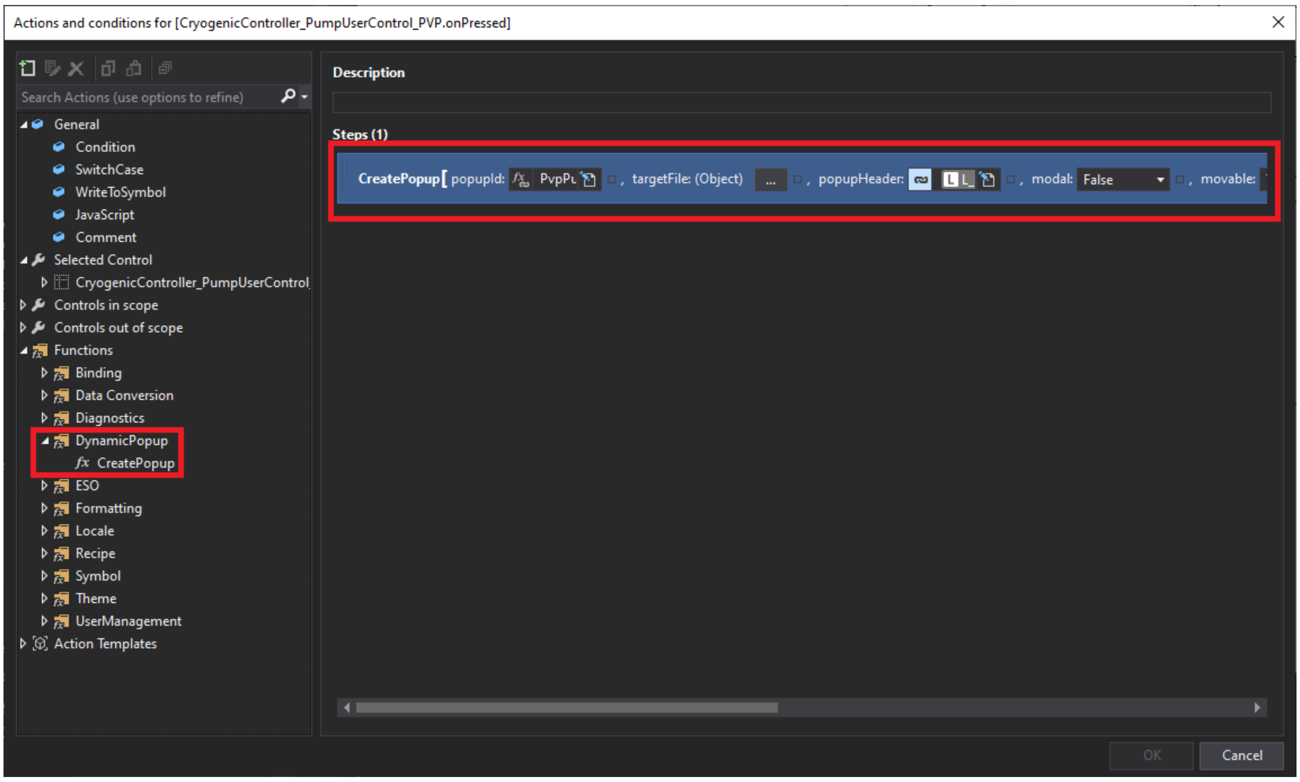


Pump State	OFF		
Control Mode	UNKNOWN		
Pump Hours	0	Ready	☐
Pump Speed	0 rpm	Running	☐
Pump Temp	0 °C		
Motor Current	0 mA		
Motor Voltage	0 V		

Fig. 11: Faceplate example of a device named PVP. Currently on Tab **Status**

parameter. When clicking on the three dots ... for the targetFile, it's necessary to fill in the faceplate that you want to be displayed and some additional parameters.

The $\mathrm{PumpObject}$ parameter is an instance of the HMI gateway function block (in this case, an instance of the $\mathrm{FB\_HMI\_CRYO\_PUMP}$). The $\mathrm{EventGrid}$ parameter refers to the (only) instance of the EventGrid control in your application. In the template, this is simply called $\mathrm{Events\_TcHmiEventGrid}$. The drop-down menu should only show one of them and this is the one that needs to be selected. The reason this is necessary to provide is because the faceplates provide a filtering possibility for events, and thus need a handle to the event grid. The $\mathrm{EventPageIdentity}$ parameter is the value of the $\mathrm{PageIdentity}$ internal symbol representing the page for events. This is necessary to have as using the event filters in the face plate will automatically open up the page of the events.

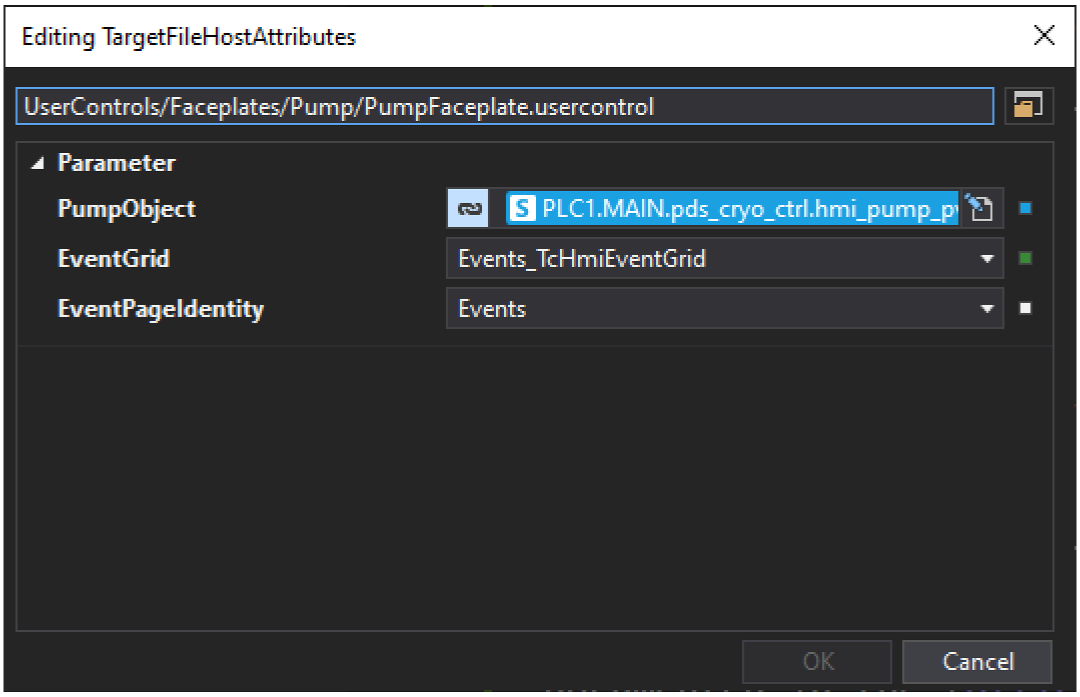

Actions and conditions for [CryogenicController_PumpUserControl_PVP.onPressed]

Description

Steps (1)

CreatePopup[popupId: PvpPu , targetFile: (Object) ... , popupHeader: L L , modal: False , movable:

Fig. 12: Pop-Up configuration of *onPressed()*-event.


Editing TargetFileHostAttributes

UserControls/Faceplates/Pump/PumpFaceplate.usercontrol

Parameter	
PumpObject	PLC1.MAIN.pds_cryo_ctrl.hmi_pump_p
EventGrid	Events_TcHmiEventGrid
EventPageIdentity	Events

Fig. 13: $\mathrm{targetFiles}$-parameter editing.

CoolerFaceplate

This is a faceplate for the stirling cooler user control.

Table 10: CoolerFaceplate Interface Table

Parameter	Description
CoolerObject	Linked to an instance of the HMI gateway function block FB_HMI_CRYO_COOLER.
EventGrid	The reference to the instance of the EventGrid control in the application
EventPageIdentity	The value of the PageIdentity internal symbol representing the page for events

PumpFaceplate

This is a faceplate for the pump user control.

Table 11: PumpFaceplate Interface Table

Parameter	Description
PumpObject	Linked to an instance of the HMI gateway function block FB_HMI_CRYO_PUMP.
EventGrid	The reference to the instance of the EventGrid control in the application.
EventPageIdentity	The value of the PageIdentity internal symbol representing the page for events

ValveFaceplate

This is a faceplate for the valve user control.

Table 12: ValveFaceplate Interface Table

Parameter	Description
ValveObject	Linked to an instance of the HMI gateway function block FB_HMI_CRYO_VALVE
EventGrid	The reference to the instance of the EventGrid control in the application.
EventPageIdentity	The value of the PageIdentity internal symbol representing the page for events

HeaterFaceplate

This is a faceplate for the heater user control.

Table 13: HeaterFaceplate Interface Table

Parameter	Description
HeaterObject	Linked to an instance of the HMI gateway function block FB_HMI_CRYO_HEATER
EventGrid	The reference to the instance of the EventGrid control in the application.
EventPageIdentity	The value of the PageIdentity internal symbol representing the page for events

PdsHeaterFaceplate

This is a faceplate for the PDS heater user control, specifically done for the PDS application, which the PLC example project is based on.

Table 14: PdsHeaterFaceplate Interface Table

Parameter	Description
HeaterObject	Linked to an instance of the HMI gateway function block FB_HMI_PDS_CRYO_HEATER
EventGrid	The reference to the instance of the EventGrid control in the application.
EventPageIdentity	The value of the PageIdentity internal symbol representing the page for events

LakeshoreFaceplate

This is a faceplate for the lakeshore controller user control.

Table 15: LakeshoreFaceplate Interface Table

Parameter	Description
LakeshoreObject	Linked to an instance of the HMI gateway function block FB_HMI_CRYO_LAKESHORE
EventGrid	The reference to the instance of the EventGrid control in the application.
EventPageIdentity	The value of the PageIdentity internal symbol representing the page for events

The lakeshore faceplate looks slightly different than the standard faceplate described above. It doesn't have the service tab as it's not possible to control the lakeshore controller from the HMI. Because of this the operate tab looks slightly different as it doesn't have the usual controls for power on/off and switching between automatic and manual control. It also provides status information from the device.

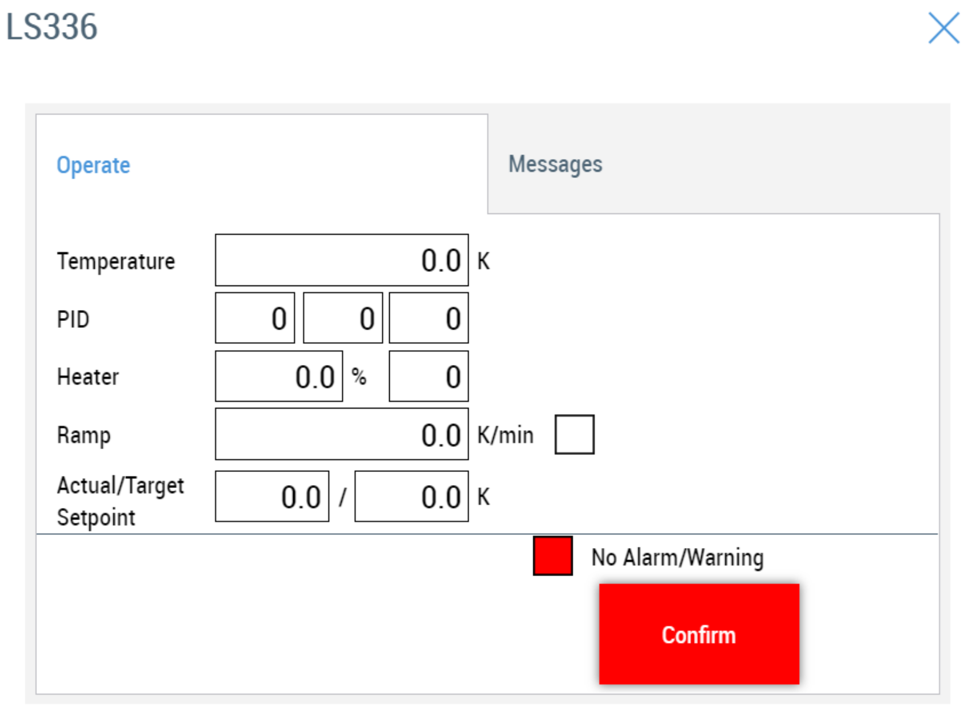


Fig. 14: Faceplate example of a Lakeshore controlled named LS336.

PressureFaceplate

This is a faceplate for the pressure user control.

Table 16: PressureFaceplate Interface Table

Parameter	Description
PressureObject	Linked to an instance of the HMI gateway function block FB_HMI_CRYO_PRESSURE_VOTE
EventGrid	The reference to the instance of the EventGrid control in the application.
EventPageIdentity	The value of the PageIdentity internal symbol representing the page for events

The pressure faceplate looks slightly different than the standard faceplate. It provides the pressure value based on the vote functionality of the FB_HMI_CRYO_PRESSURE_VOTE. Additionally, it provides the sensor diagnostics for each individual pressure sensor (up to five). Note that it only displays the pressure diagnostics of the sensors that are actually configured, i.e. sensors that are not configured will not be displayed. The faceplate gives a possibility to simulate the pressure, i.e. override the actual pressure data. This is achieved by entering the simulated pressure in the "Pressure Simulated"-textbox and by pressing the "Simulation ON"-button. To go back to the real pressure vote data, this is achieved by pressing the "Simulation OFF"-button.

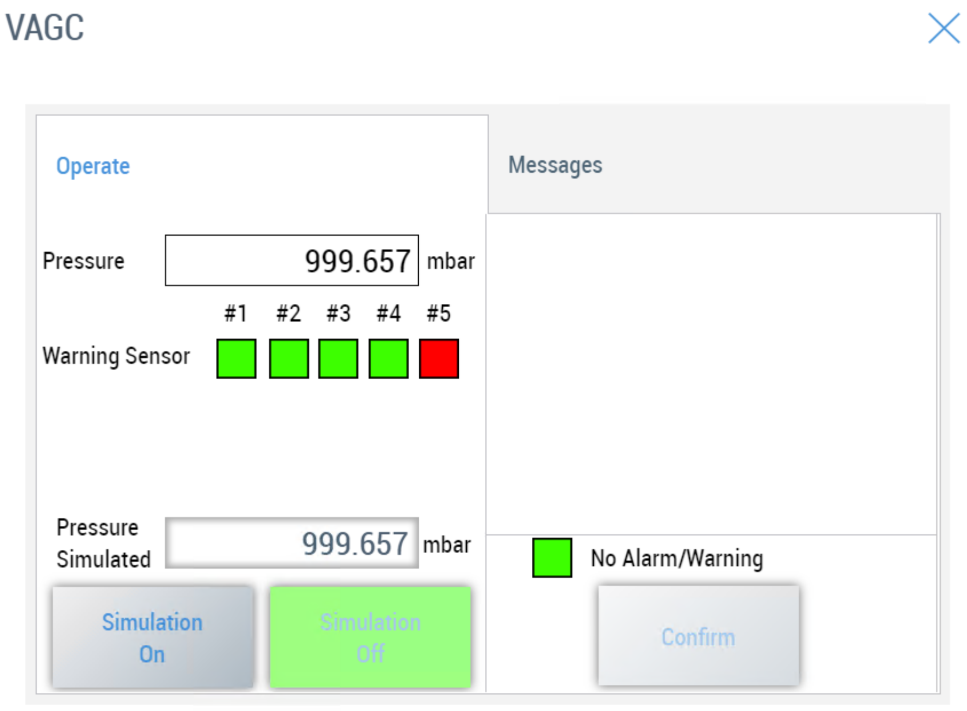


Fig. 15: Faceplate example of a pressure vote named VAGC, in non-simulated mode

TemperatureFaceplate

This is a faceplate for the temperature user control.

Table 17: TemperatureFaceplate Interface Table

Parameter	Description
TemperatureObject	Linked to an instance of the HMI gateway function block FB_HMI_CRYO_TEMPERATURE
EventGrid	The reference to the instance of the EventGrid control in the application.
EventPageIdentity	The value of the PageIdentity internal symbol representing the page for events

The temperature faceplate looks slightly different than the standard faceplate. It provides the temperature value based on the functionality of the FB_HMI_CRYO_TEMPERATURE. Additionally, it provides the sensor diagnostics for each individual temperature sensor (always two). The faceplate gives a possibility to simulate the temperature, i.e. override the actual temperature data. This is achieved by entering the simulated temperature in the “Temperature Simulated”-textbox and by pressing the “Simulation ON”-button. To go back to the real temperature data, this is achieved by pressing the “Simulation OFF”-button.

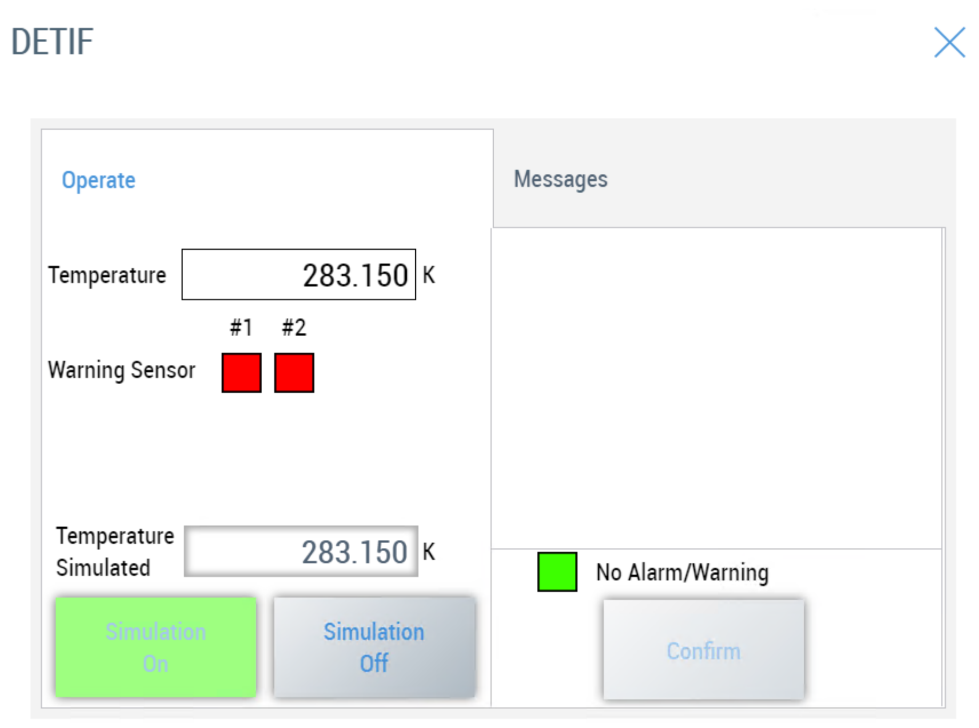


Fig. 16: Faceplate example of a temperature named DETIF, in simulated mode

Sequence faceplates

The sequence faceplate is used to give basic control of a sequence. It's implemented as an user control and can be dragged and dropped into any content.

When pressing the start or stop-button, the following is overlayed.

Only after pressing **OK** is the start/stop command executed. Pressing **CANCEL** will mean the start/stop command is not executed, and the faceplate is presented again. The purpose of this overlay is so that the operator doesn't accidentally stop/start a sequence but has to confirm the action first.

The user control consists of the following elements:

1. **The title**. The title of the faceplate. Should describe what the sequence does.
2. **The status**. Shows the current status of the sequence. This is a longer textual description of what the sequence is currently doing.
3. **Start & Stop buttons**. Allows the operator to start or stop the sequence.
4. **Permit Start & Stop indicators**. Shows whether it's possible to start or stop a sequence.
5. **Detailed sequence information**. Pressing this button will show a flowchart of the sequence.

When creating an instance of the sequence faceplate, the following parameters are available:

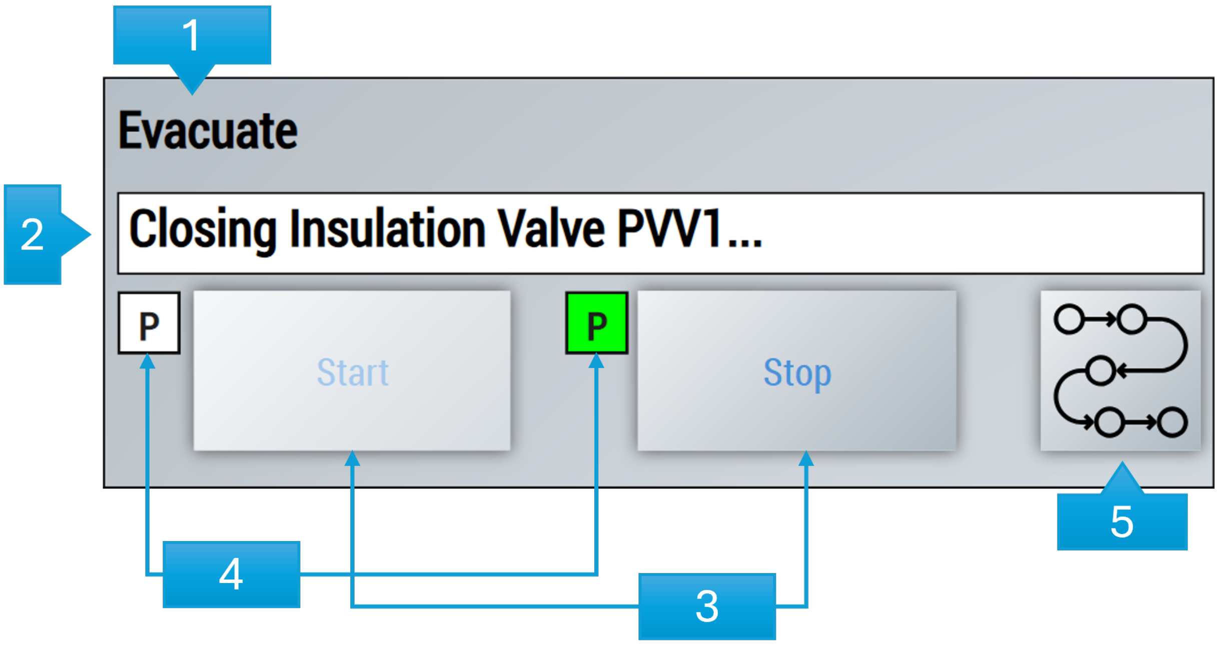


Fig. 17: The SequenceFaceplate.usercontrol

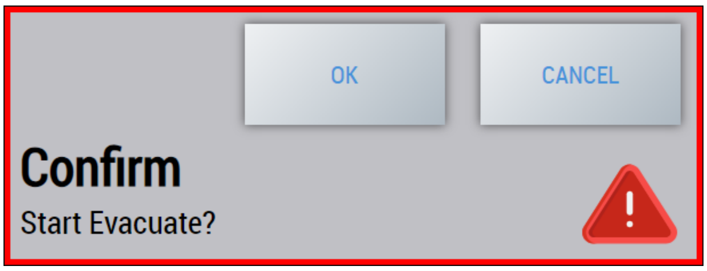


Fig. 18: Example: Confirm start of evacuation sequence

Table 18: SequenceFaceplate Interface Table

Parameter	Description
Title	The title in the faceplate. Represents (1) in the *figure above*.
SequenceState-Text	The string that represents (2) in the *figure above*. Linked to a string variable.
StartStopSe-quence	The boolean variable to toggle the start/enable (true) or stop/disable (false) of the sequence.
SequencePageI-dentity	The value of the PageIdentity internal symbol representing the page for the sequence.
ShowSequence-Button	Set to true if sequence button should be visible, otherwise set to false. Useful for example if you want to be able to show the sequence faceplate on the actual sequence page (like for example the EvacuateSequence.content) for controlling the sequence (stop/start) but don't want that the sequence button to be visible.

For the button (5) to work in the *figure above*, it's necessary to:

1. Create a *separate content-page* where the detailed sequence will be shown.
2. Add that page as a value in the PageIdentity data type (available under **TwinCAT HMI Configuration $\rightarrow$ Data types**).
3. Set the parameter SequencePageIdentity of the instance of the SequenceFaceplate.usercontrol to the new value created in the previous step.
4. Open up the ControlCabinet.view, locate the TcHmiRegion with identity ControlCabinet_TcHmiRegion_Main, go to the custom event listener SelectedPageIdentity and add the new content to the switch-case. If the ControlRoom.view is used, do the same for it.

By doing these steps, the button *(5)* will display the new content-page when it's pressed.

3.3 Sequence

The details of how sequences work are described in the chapter **Sequences** of the Cryo HMI framework documentation. This chapter will focus on the usage of the sequences in the actual HMI application.

When using the *sequence faceplate*, for the button (5) to work in the *figure above* it's necessary to create a separate content page where the detailed sequence will be displayed in. The EvacuateSequence.content is the example for this in the TcHmiTemplate project. This is a standard TwinCAT HMI content page. To get a fully working sequence, it's necessary to add the following to the content:

1. An instance of TcHmi.Controls.CryoFramework.SequenceContainer from the Cryo HMI Framework. The SequenceContainer needs to be linked to the sequence in the PLC through an instance of the T_CRYO_HMI_SEQUENCE_STAT struct, in terms instantiated through creating an instance of the FB_CRYO_HMI_SEQUENCE function block.
2. A combination of TcHmi.Controls.CryoFramework.Process, TcHmi.Controls.CryoFramework.Decision and TcHmi.Controls.CryoFramework.Terminator to define the sequence. These

have to be put inside the SequenceContainer, and assigned an unique BlockId (parameter State → BlockId) representing their location in the array arSequence located in T_CRYO_HMI_SEQUENCE_STAT. Make sure all elements are located in the SequenceContainer by using the document outline (TwinCAT HMI → Windows → TwinCAT HMI Document Outline) and checking that all elements are sub-nodes under the SequenceContainer.

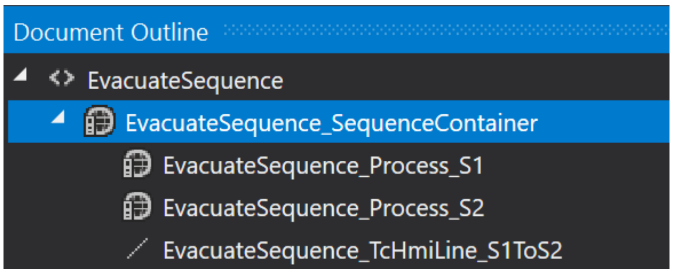
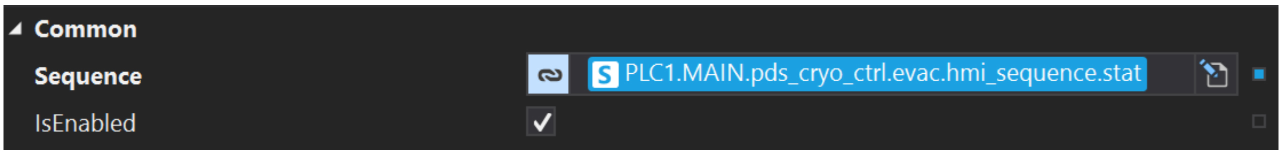

Document Outline
- EvacuateSequence
  - EvacuateSequence_SequenceContainer
    - EvacuateSequence_Process_S1
    - EvacuateSequence_Process_S2
    - EvacuateSequence_TcHmiLine_S1ToS2

Fig. 19: TwinCAT HMI document outline: Sequence Container

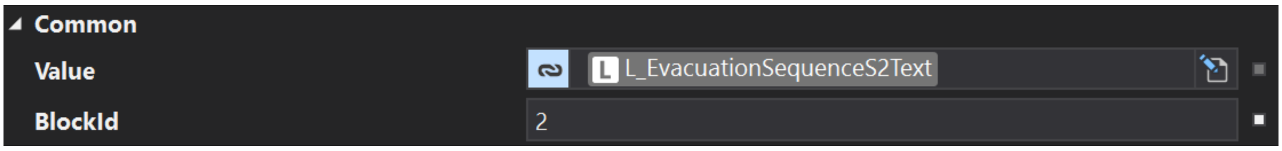
The SequenceContainer example in the TcHmiExample project is linked to an instance of T_CRYO_HMI_SEQUENCE_STAT with the symbolic name PLC1.MAIN.pds_cryo_ctrl.evac.hmi_sequence.stat:


Common

Sequence	PLC1.MAIN.pds_cryo_ctrl.evac.hmi_sequence.stat
IsEnabled	✓

Fig. 20: TwinCAT HMI document outline: Sequence Container linking

Every Process, Decision and Terminator has to have an unique BlockId number, equivalent to the number that is used to define that particular step in the sequence in the PLC-code.


Common

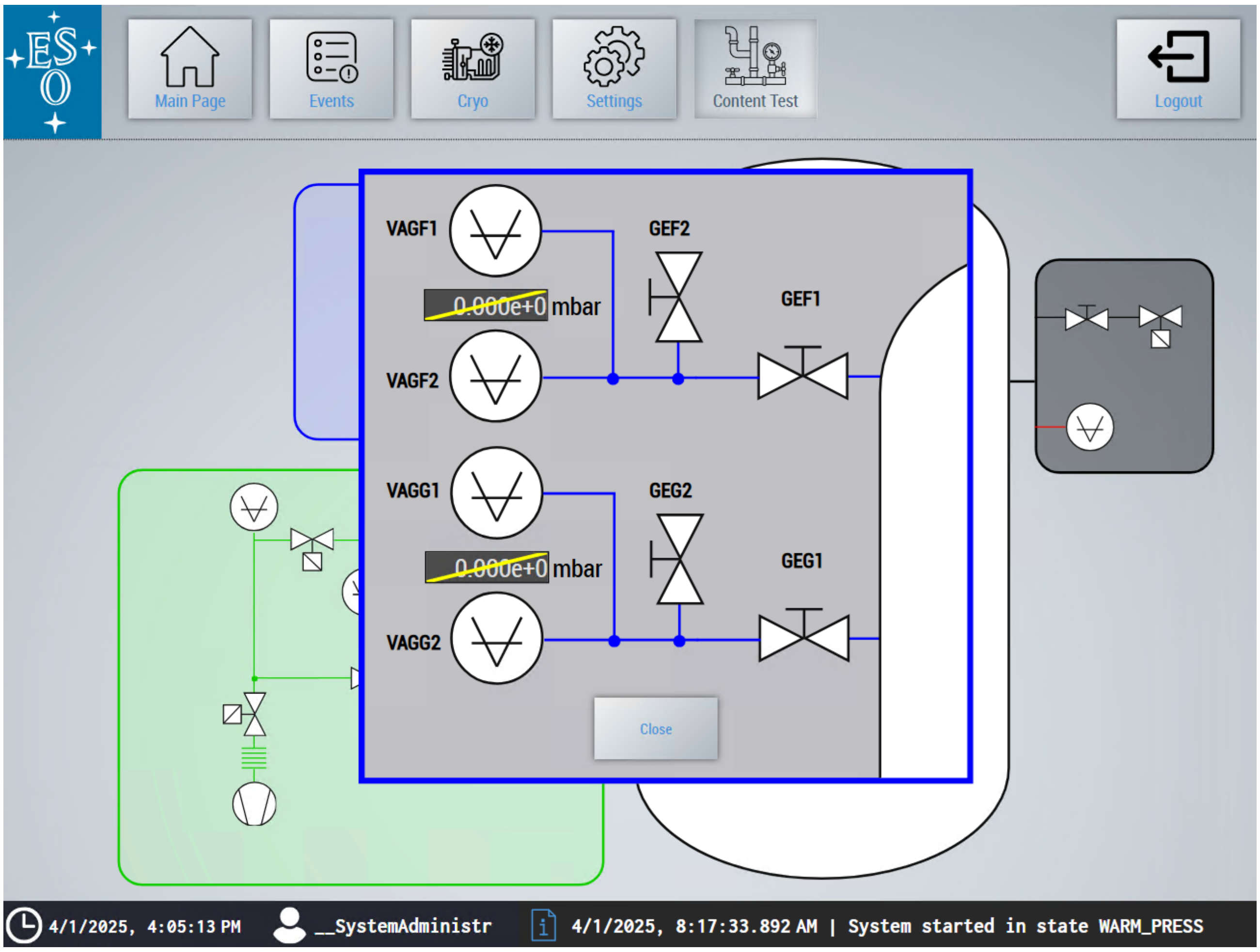
Value	L_EvacuationSequenceS2Text
BlockId	2

Fig. 21: Sequence Container Node: BlockId linking

3.4 Content test

When running the HMI on a small screen size (such as a 15”), there might be situations where there is not enough space on the screen to fit all the controls for a system. To demonstrate one way for a work-around there is a page called “Content test” in the template, which demonstrates the utilization of the screen in such a way that more content can be presented.

By clicking on any of the green, blue or black areas of the page, a user control is shown with more details for that area of the system. In this way it’s possible to utilize the full screen for a part of the

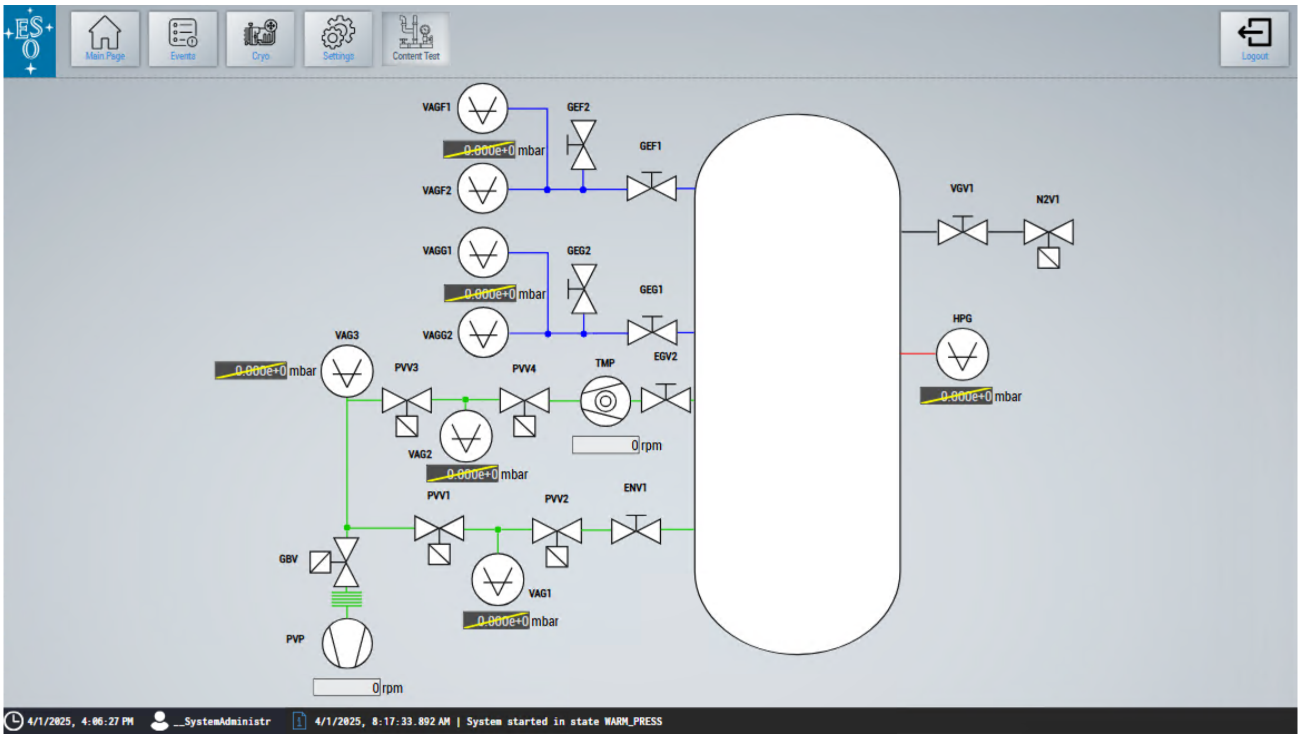


4/1/2025, 4:05:13 PM __SystemAdministr 4/1/2025, 8:17:33.892 AM | System started in state WARM_PRESS

Fig. 22: Content test: Control cabinet

system.

In situations where there is more space available (such as on a typical office monitor), the ControlRoom-view is used in where all the content is displayed in one page without the need for a "zoom-in" view. See *Views* for more information about the two available views and how to switch between them.



4/1/2025, 4:06:27 PM __SystemAdministr 4/1/2025, 8:17:33.892 AM | System started in state WARM_PRESS

Fig. 23: Content test: Control room

As usual the controls can be clicked-on to display the popups for controlling the user controls. Note that this page is for pure demonstration, and thus no integration to PLC variables is made.