TIA Portal Openness: Introduction and demo application

TIA Portal V17.0

https://support.industry.siemens.com/cs/ww/en/view/108716692

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.
Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.
By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.
The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.
Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under https://www.siemens.com/cert.

Table of contents

Legal information **2**

1 Task **5**

2 Solution **6**

2.1 Overview 6

2.2 Hardware and software components 7
2.2.1 Applicability 7
2.2.2 Components used 7

3 Mode of operation **8**

3.1 TIA Portal Openness 8
3.1.1 Feature set 8
3.1.1.1 Dependencies 9
3.1.2 Limitations 9

3.2 "Basic Project Generator" example 9

3.3 "TIA Portal Openness Demo" example 9

4 Creating a new TIA Portal Openness application **10**

4.1 TIA Portal V17.0 10

4.2 Managing user rights 10

4.3 Creating the project 12

4.4 Configuration file / AssemblyResolve 13

4.5 Grant access 13

5 Basic Project Generator **14**

5.1 Overview 14

5.2 "TIA Portal" group 17

5.3 "TIA Portal project" group 18

5.4 "Add new device" group 22

5.5 "Compile" group 26

6 TIA Portal Openness Demo Application **28**

6.1 Overview 28

6.2 Setting up the solution 29
6.2.1 Unload project 29
6.2.2 Reload project 30
6.2.3 Exclude project folder from a project 31
6.2.4 Include a project folder in a project 32

6.3 Assembly Resolve 40

6.4 Preselection Modules 41

6.5 Main Window 42

6.6 "TIA Portal" slide panel 44

6.7 Slide Panel "Libraries" 45
6.7.1 Project library 45
6.7.1.1 Creating or editing groups 46
6.7.1.2 Copying a type version from the project library to the project 47

6.7.1.3 Exporting type versions....48
6.7.2 Global libraries....49

6.8 "File" menu....51
6.8.1 Open TIA Portal....51
6.8.2 Close TIA Portal....51
6.8.3 Connect TIA Portal....52
6.8.4 Disconnect TIA Portal....53
6.8.5 Open local Session....53
6.8.6 Save local Session....54
6.8.7 Close local Session....54
6.8.8 Create Project....54
6.8.9 Open Project....55
6.8.10 Save Project....56
6.8.11 Close Project....56

6.9 "View" menu....57
6.9.1 Logical view....58
6.9.2 Physical view....59

6.10 "Project" menu....61
6.10.1 Create new group....61
6.10.2 "TIA Portal editor" submenu....62
6.10.2.1 Open Editor....63
6.10.2.2 Topology view....64
6.10.2.3 Network view....65
6.10.3 Compile....66
6.10.4 "Import/Export" submenu....68
6.10.4.1 CAx Import....68
6.10.4.2 CAx Export....69
6.10.4.3 Import Element as Simatic ML....70
6.10.4.4 Import Structure as Simatic ML....71
6.10.4.5 Export Structure as Simatic ML....75

6.11 "PLC" menu....77
6.11.1 Add external source....77
6.11.2 Generate blocks from external source....79
6.11.3 Generate source from block....81

6.12 "Options" menu....84
6.12.1 Settings....84

6.13 "Help" menu....86
6.13.1 About TIA Portal Openness Demo....86

7 Appendix....87

7.1 Service and support....87

7.2 Industry Mall....88

7.3 Links and literature....88

7.4 Change documentation....88

1 Task

Introduction

In STEP 7 V17.0, WinCC Unified V17.0 or WinCC Professional V17.0, referred to from now on as WinCC V17.0, TIA Portal Openness is included in the scope of delivery of STEP 7 V17.0 / WinCC 17.0 in TIA Portal. This enables you to program applications which automate engineering in TIA Portal.

Note Please note that the feature set of TIA Portal Openness depends on the base installation of STEP 7 17.0 or WinCC 17.0.

Overview of the automation task

Scenarios

- Making use of a text database, project texts must be automatically translated by the program and pushed to the TIA Portal project. This allows you to quickly use standardized texts in new projects.
- The visualization must be automatically created using exported PLC data.
- Project statistics or backups can be generated automatically. Using your program, you can check whether programming guidelines have been adhered to.
- Projects can be automatically compared against global libraries and, if needed, updated and compiled.
- The offline project can be automatically compared with the online projects to ensure plant consistency.
- Tools and prefabricated project components are used to create whole projects.

2 Solution

2.1 Overview

Advantages

The solution presented here offers you the following advantages:

- Increased efficiency, thanks to faster execution of the task
- Accuracy, thanks to the automation of consistent processes
- Short commissioning times, thanks to program-supported creation of configurations
- Competitiveness, thanks to targeted use of resources

Delimitation

This application example does not provide a description of:

- Basics of object-oriented programming
- Basis on the programming environment, such as Microsoft Visual Studio
- Basics of TIA Portal configuration

Basic knowledge of these topics is required.

2.2 Hardware and software components

2.2.1 Applicability

These application examples are valid for

- STEP 7 V17.0 / WinCC V17.0 TIA Portal V17.0 TIA Portal Openness V17.0

2.2.2 Components used

The application examples were created with the following components:

Software components

Table 2-1

Component	Qty.	Item number	Note
STEP 7 V17.0	1	6ES7822-1A.07-..	
WinCC V17.0	1	6AV210.-....7-0	
TIA Portal Openness V17.0	1		Included in delivery of STEP 7 V17.0 or WinCC V17.0
Microsoft Visual Studio 2019	1		

Sample files and projects

The following Table provides a list of all files and example projects.

Table 2-2

Component	Note
108716692_TIA_PortalOpenness_ GettingStarted_V17.zip	Introductory example on using TIA Portal Openness (see chapter 5).
108716692_TIA_PortalOpenness_ Demo_V17.zip	Comprehensive example on using TIA Portal Openness (see chapter 6).
108716692_TIA_PortalOpenness_ GettingStartedAndDemo_V17_en.pdf	This document.

3 Mode of operation

3.1 TIA Portal Openness

TIA Portal Openness is included for free on the respective product DVDs for STEP 7 V17.0 / WinCC V17.0. Its use requires that STEP 7 V17.0 or WinCC V17.0 first be installed. TIA Portal Openness provides you with DLLs that you can use to access the TIA Portal platform. These DLLs are based on .NET Framework 4.6.1.

3.1.1 Feature set

Table 3-1

Scope of application	Function	Further information / limitations
Project	Open TIA Portal	
	Close TIA Portal	
	Connect TIA Portal	
	Disconnect TIA Portal	
	Open local Session	Available in this form as of V17.0.
	Save local Session	
	Close local Session	
	Create Project	
	Open Project	
	Save Project	
	Close Project	
View	Logical Tree	see Dependencies
	Physical Tree	see Dependencies
	Create new group	see Dependencies
	Open editor	
	Topology view	
	Network view	
	Compile	
	CAx Import	
	CAx Export	
	Import Element as Simatic ML	see Dependencies
	Export Structure as Simatic ML	see Dependencies
PLC	Add external source	
	Generate blocks from external source	
	Generate source from block	
Options	Settings	
Help	About TIA Portal Openness Demo	
Project library	Create new group	see Dependencies
	Edit group name	see Dependencies
	Copy element to project	see Dependencies

Scope of application	Function	Further information / limitations
	Export version	
Global library	Create new user library	
	Open user library	
	Save library	
	Close user library	
	Update project library	
	Export version	

3.1.1.1 Dependencies

Availability, content and scope depend on your version as well as on the installed software modules, such as STEP 7, WinCC Unified, WinCC Professional and Sinamics Startdrive. See Figure 6-8.

3.1.2 Limitations

- With TIA Portal Openness V17.0, you can only access projects and libraries with Version V17.0. If necessary, upgrade your project and/or your library before use with TIA Portal Openness.
- No compatibility can be guaranteed between the different versions of TIA Portal Openness. With a new version, changes to your program may be necessary.

3.2 "Basic Project Generator" example

The "Basic Project Generator" program is intended to help you get started with programming your first Openness application.

Some basic functions are already programmed into the program (e.g. open TIA Portal), allowing you to build your own application on top of it.

3.3 "TIA Portal Openness Demo" example

The "TIA Portal Openness Demo" program contains many fully-programmed functions of TIA Portal Openness. It is intended to offer you an overview of the functionality as well as being a detailed programming aid.

4 Creating a new TIA Portal Openness application

4.1 TIA Portal V17.0

Install TIA Portal V17.0.

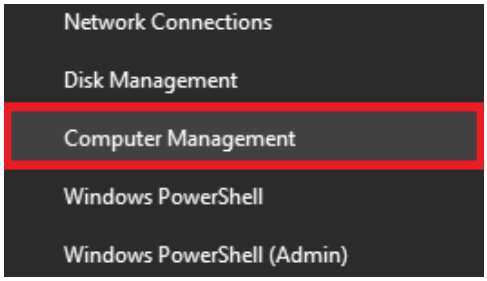
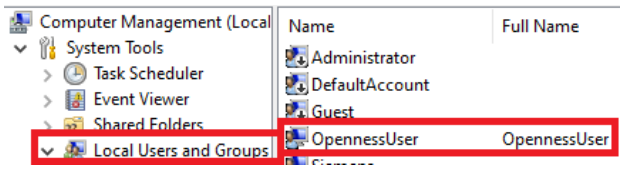
Note

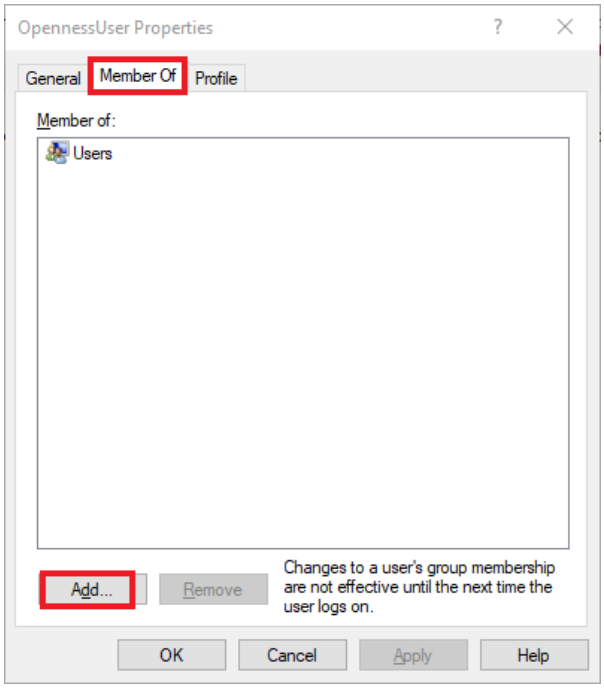
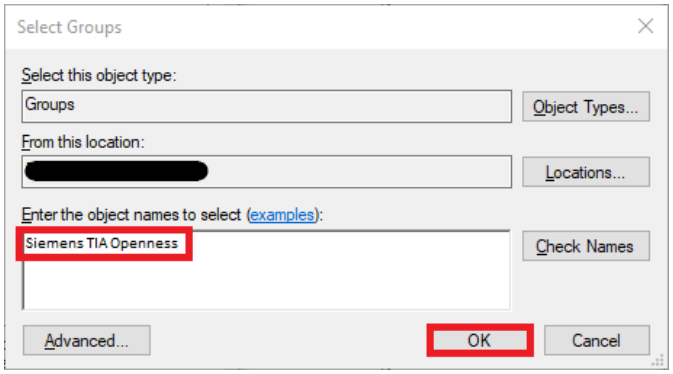
In STEP 7 V17.0 or WinCC V17.0, TIA Portal Openness V17.0 is included in the scope of delivery and is installed along with it by default.

4.2 Managing user rights

To use and/or create the TIA Portal Openness application, the user must be added to the "Siemens TIA Openness" user group.

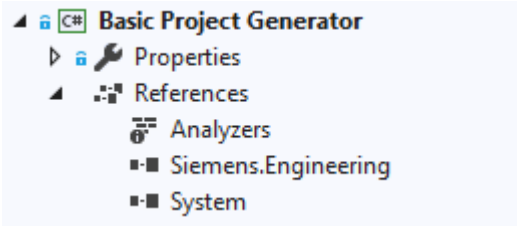
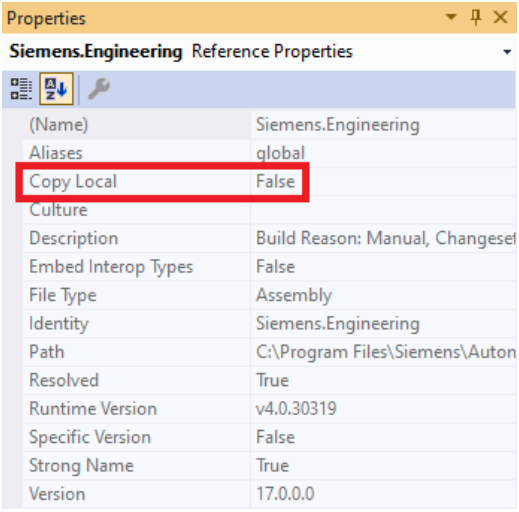
Table 4-1

No.	Action
1.	Right-click on the Start (Windows) icon in the Windows taskbar. Select "Computer Management" and confirm the UAC dialog message by clicking "OK". 
2.	Open "Local Users and Groups > Users", and double-click on the username "OpennessUser". 

No.	Action
3.	Change to the "Member Of" tab and click the "Add..." button. 
4.	Enter "Siemens TIA Openness" and confirm by clicking "OK". 
5.	Close all open dialogs and log in again.

4.3 Creating the project

Table 4-2

No.	Action
1.	Create a new project (e.g. in Microsoft Visual Studio).
2.	Create the references to the Openness DLLs (Siemens.Engineering.dll und Siemens.Engineering.HMI.dll). They are located in the TIA Portal installation directory under "... > Siemens > Automation > Portal V17_0 > PublicAPI > V17.0".
3.	Set the "Copy Local" property of both DLLs to "False".  

4.4 Configuration file / AssemblyResolve

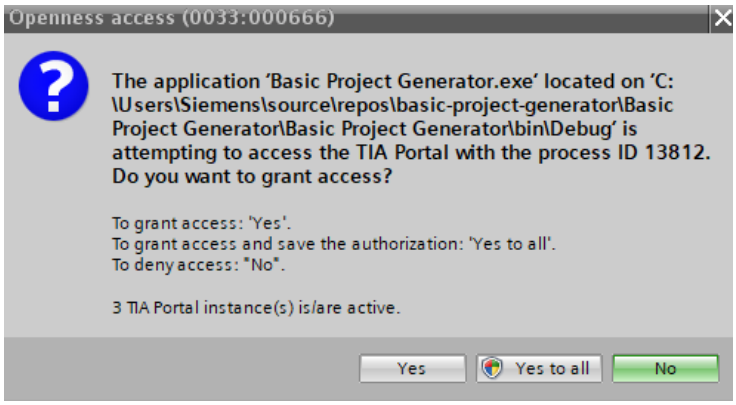
To find the path to the Openness DLLs, you can use either a configuration file or the "AssemblyResolve" event.

Table 4-3

No.	Action
1.	**Configuration file** If you chose a different path than the default path when installing STEP 7 V17.0 or WinCC V17.0 (TIA Portal), replace the default path with your installation path in the configuration file. Create the application configuration file in the same directory as the Openness application.
2.	**AssemblyResolve** To establish the connection to TIA Portal, you can use the `Resolver.GetAssemblyPath` method. This involves reading the TIA Portal installation path from the registry so that the program can be used independently of the installation path.

4.5 Grant access

Table 4-4

No.	Action
1.	The following security notification appears when you launch the application for the first time:  Source: System manual (https://support.industry.siemens.com/cs/ww/en/view/109477163)
2.	Confirm the message with "Yes" to allow access once. Confirm the message with "Yes, all" to always allow access for this application. Click "No" to deny access.

Note

If you are working with Microsoft Visual Studio, you might receive the message even though you have already clicked "Yes to all". Follow the instructions in the article referenced in \5\ to prevent this.

5 Basic Project Generator

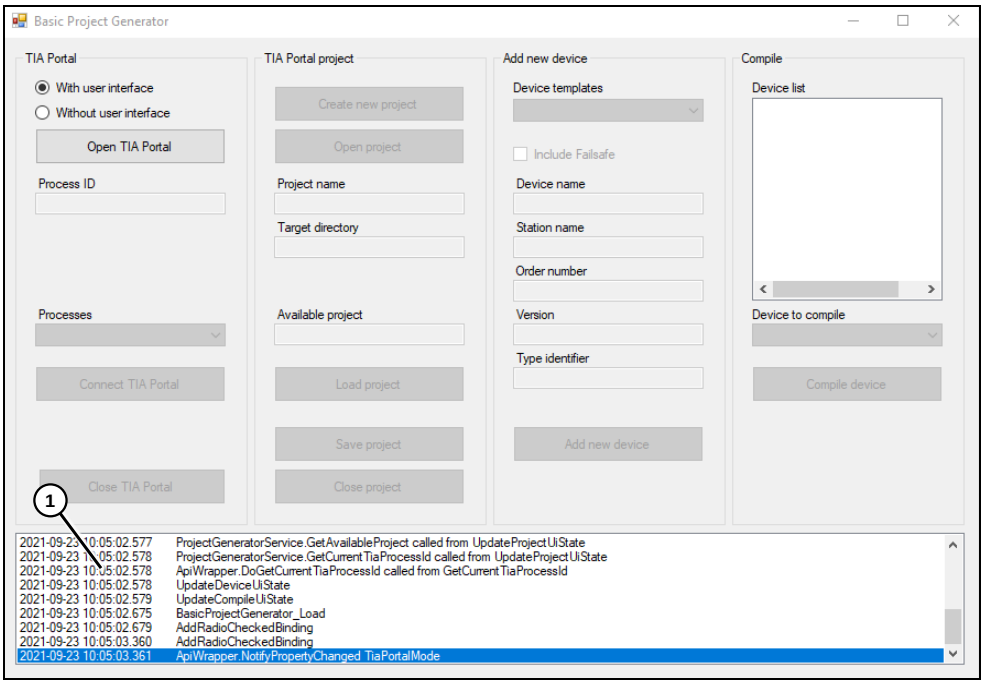
5.1 Overview

The "Basic Project Generator" program is intended to help you get started with programming your first Openness application.

Note

You can find a fully compiled "exe" file in the download "108716692_TIA_PortalOpenness_GettingStarted_V17.zip" in the "BasicProjectGenerator_Appl.zip"

Figure 5-1



Status information about the program sequence appears in the list box. The current status is highlighted in blue.

Figure 5-2

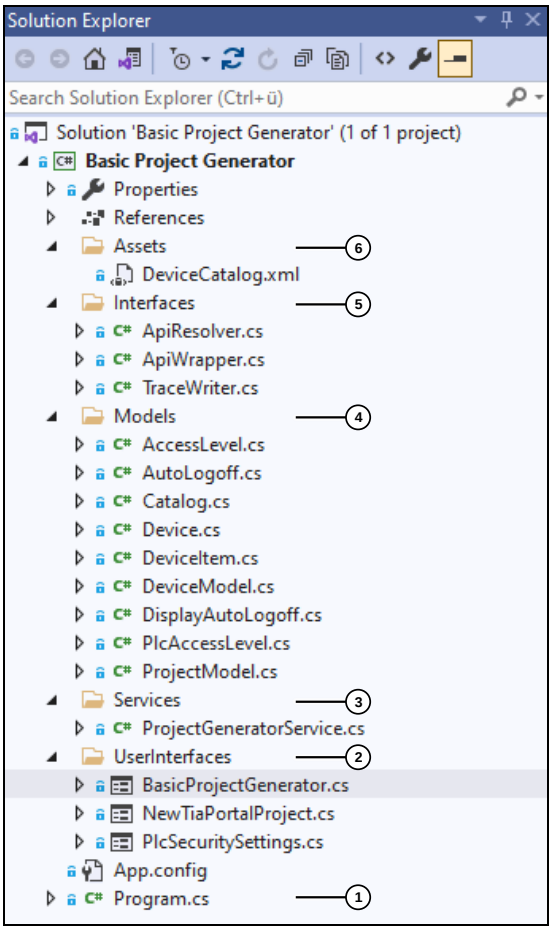


Table 5-1

No.	Description
1.	`Program.cs` contains the Main function as a jumping-off point for the application.
2.	The `BasicProjectGenerator.cs`, as the main view of the application (see Figure 5-1) contains all control elements and the implementation of the event handler for the control elements. `NewTiaPortalProject.cs` is a dialog view for creating new projects. `PlcSecuritySettings.cs` is a dialog view for adding security settings when creating new devices.
3.	`ProjectGeneratorService.cs` is the interface between the user interface (see item 2 in this Table) and the TIA Portal Openness API (see item 5 in this Table).
4.	All data objects are defined in the `Models` area. A detailed description (if necessary) can be found in the chapters pertaining to the groups.
5.	In the `Interfaces` area you can find the implementation of the access logic to the TIA Portal Openness API in the form of `ApiWrapper.cs` as well as interfaces/services used across the whole application. `ApiWrapper.cs` is the only class that contains a reference to "Siemens.Engineering.dll". All API function calls are grouped in regions according to topic in the `ApiWrapper` class.
6.	`DeviceCatalog.xml` is a template file that provides meta-information about adding new devices. New devices can be added to this device catalog at any time. The device catalog is reloaded when creating or opening a project. This is done by calling the method `LoadDeviceCatalog` in the class `ProjectGeneratorService` .

5.2 "TIA Portal" group

Figure 5-3

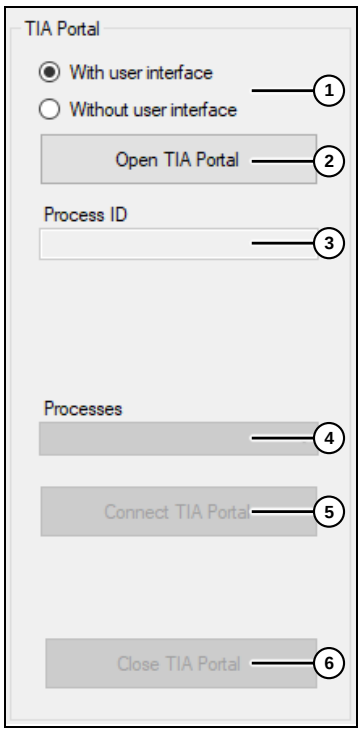


Table 5-2

No.	Description
1.	Use the `With user interface` and `Without user interface` radio buttons to select whether to launch TIA Portal with or without a user interface. In the `BasicProjectGenerator_Load` method, the radio buttons are linked to the `TiaPortalMode` property of the `ApiWrapper` class. They are assigned to the values that must be set when an option is selected.
2.	To open an instance of TIA Portal, click the `Open TIA Portal` button. The `DoOpenTiaPortal` method of the `ApiWrapper` class creates a new instance and assigns it to the `TiaPortal` property of the `ApiWrapper` class. Because the `TiaPortalMode` property is linked to the radio buttons (see item 1), the selected start mode will be used automatically. `TiaPortal = new TiaPortal(TiaPortalMode);`
3.	After creating a TIA Portal instance, the process ID of this instance will appear in the text field as information (see Figure 5-7).
4.	After the application starts, all TIA Portal instances will be detected and, together with the `TIA Portal Mode` information, made available as a list in the Combo Box (see Figure 5-7). Once a process ID is selected that differs from the process ID in item 3, the `Connect TIA Portal` button will be automatically activated.
5.	Click the `Connect TIA Portal` button to create a connection to a TIA Portal instance. The `DoConnectTiaPortal` method in the `ApiWrapper` class is called. The information in the text field from item 3 is updated with the process ID of the connected instance.

No.	Description
6.	Click the `Close TIA Portal` button to close the TIA Portal instance. In the class `ApiWrapper`, the method `DoCloseTiaPortal` is called. Any project still open will be closed in the process. If the project contains unsaved changes, you will be prompted to pick an option (see Figure 5-8).

5.3 "TIA Portal project" group

Figure 5-4

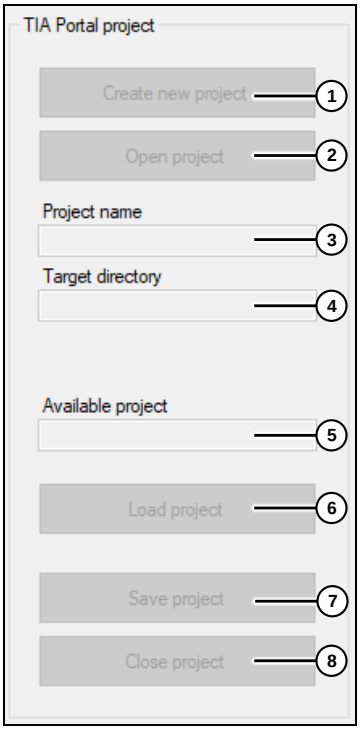


Table 5-3

No.	Description
1.	If you want to create a new TIA Portal project, click the `Create new project` button and follow the instructions in Table 5-4. First, the `CreateNewProject` method in the `ProjectGeneratorService` class is called in order to enter the project name and the destination directory. Then the system calls the method `DoCreateNewProject` in the class `ApiWrapper`.
2.	If you want to open an existing TIA Portal project, click the `Open project` button and follow the instructions in Table 5-5. The method `DoOpenProject` is called in the class `ApiWrapper`.
3.	The name of the open project appears as information in the `Project name` text field.
4.	The path of the open project file appears as information in the `Target directory` text field.
5.	If a connection was established with a running TIA Portal instance which already contained an open project, then the project name appears in the text field `Available project` and the `Load project` button is automatically enabled.

No.	Description
6.	If you want to display or edit the project data from an available project (see item 5), click on the `Load project` button. The method `DoLoadProject` is called in the class `ApiWrapper`.
7.	If you made changes to a project, the `Save project` button will be automatically enabled. Clicking on the `Save project` button will save all changes to the project. The method `DoSaveProject` is called in the class `ApiWrapper`.
8.	Click the `Close project` button to close an open project. This will run a check for whether the project was modified. If yes, you must make a decision before closing (see Figure 5-8). In that case, follow the instructions in Table 5-6. The method `DoCloseProject` is called in the class `ApiWrapper`.

Figure 5-5

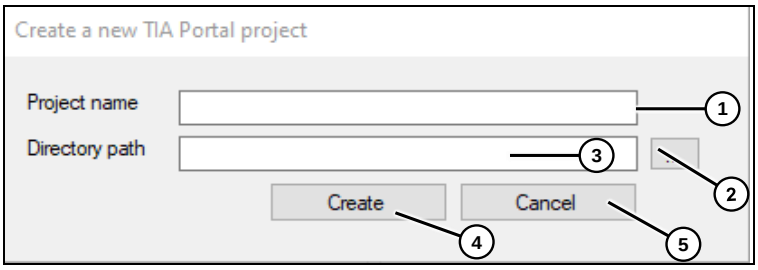


Table 5-4

No.	Description
1.	Enter a name for the new project. The name must comply with the Windows rules for file names.
2.	Open the File Explorer and select the destination directory where the new project will be created.
3.	The selected directory path appears in the text field and can be edited there if desired.
4.	Click the "Create" button to create the new project. By calling `ValidationProvider`, your input will first be validated and, if valid, the dialog will close and the new project will be created.
5.	Click the "Cancel" button if you wish to abort the process.

Figure 5-6

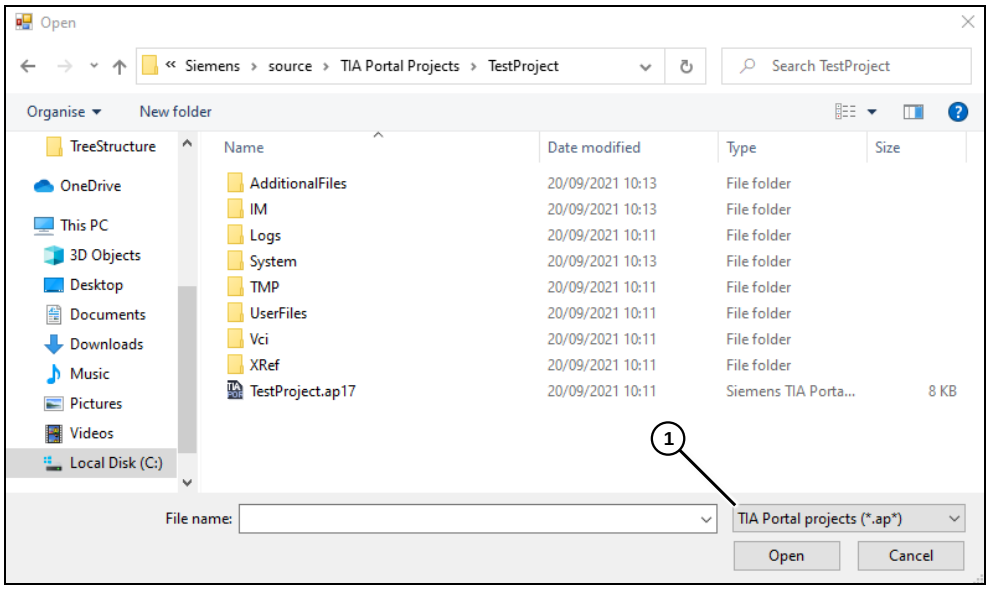


Table 5-5

No.	Description
1.	The file filter for the project files is set to `*.ap`, and so all project versions will appear. Make sure to load the correct version.

Figure 5-7

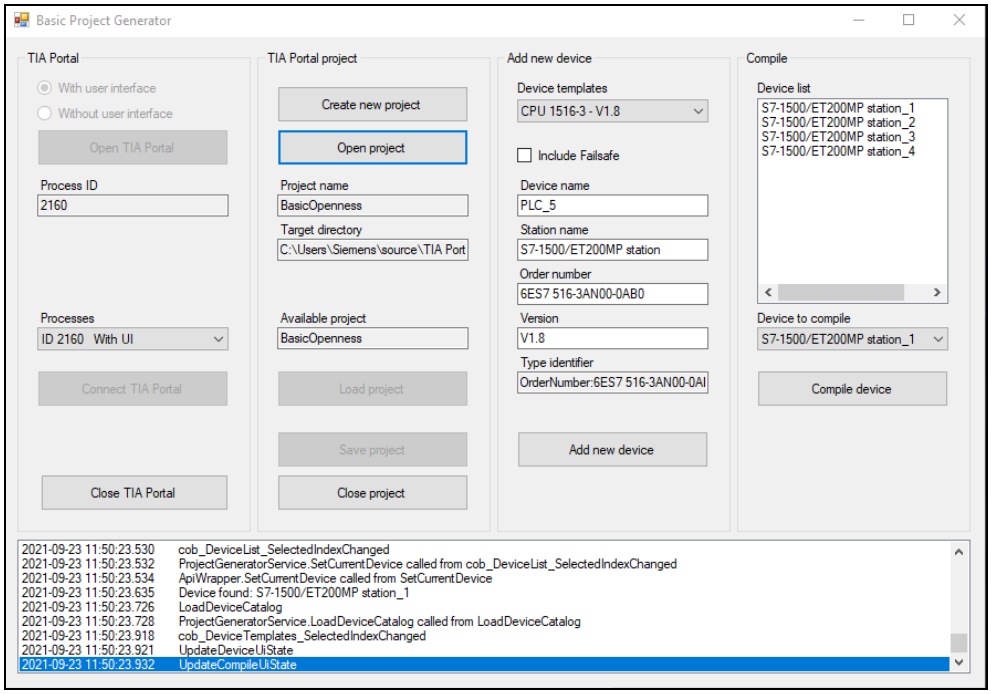


Figure 5-8

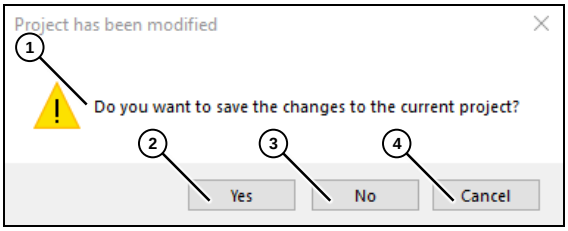


Table 5-6

No.	Description
1.	The message box (see Figure 5-8) will prompt you to decide what will happen with the changes to the project.
2.	To save the changes and close a project, click "Yes".
3.	Click "No" to close a project without saving.
4.	Click "Cancel" to abort the "closing process".

5.4 "Add new device" group

Figure 5-9

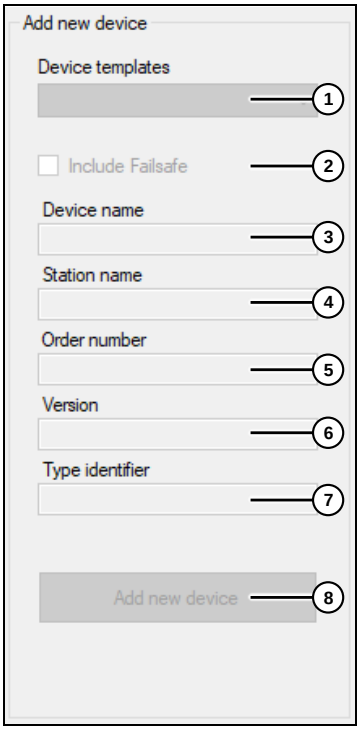


Table 5-7

No.	Description
1.	When creating a new project or opening an existing project, the device catalog (see Table 5-1 item 6) is loaded and made available as a device list in the Combo Box (see Figure 5-10).
2.	The device catalog provides the metadata `Include Failsafe`, which is enabled or disabled when selecting a device template from the list. This depends on how the template was configured. If the `Include Failsafe` option is set, then the Access Level `Full access incl. fail-safe (no protection)` will be selectable in the Security Settings dialog. Otherwise, this Access Level will not be present.
3.	The device name from the template.
4.	The station name from the template.
5.	The order number from the template. Please be sure to observe the note below!
6.	The version here is the firmware version.
7.	The Type identifier is the property `public string TypeIdentifier => "OrderNumber:" + OrderNumber + "/" + FirmwareVersion;` from the `Device` class. It is displayed as read-only information. Please be sure to observe the note below!
8.	Click the `Add new device` button to add a new device. When the new device is added, the `Security Settings` dialog opens automatically. You will be prompted to enter the settings (see Table 5-8).

Note

Enter invalid device information such as Order number and/or Version can cause a program error (see Figure 5-12). Here, calling `DoAddNewDevice` in the `ApiWrapper` class will fail.

Therefore, only add valid device information to the device catalog.

Figure 5-10

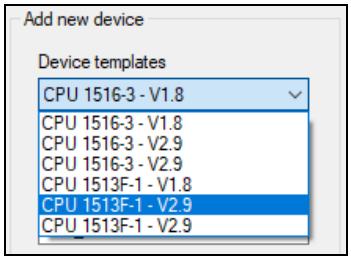


Figure 5-11

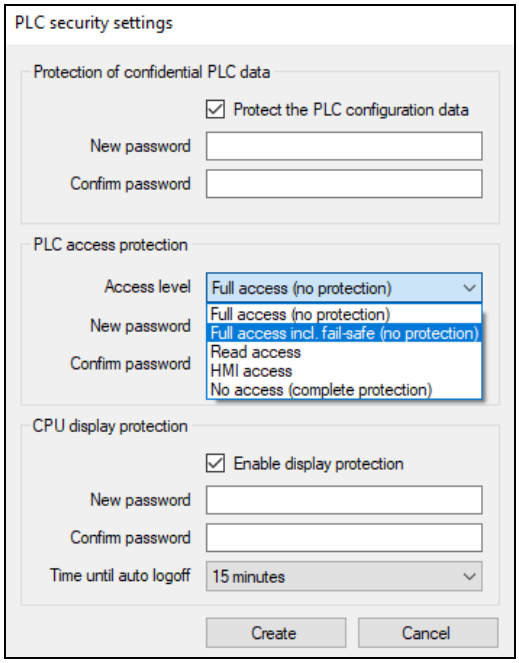


Figure 5-12

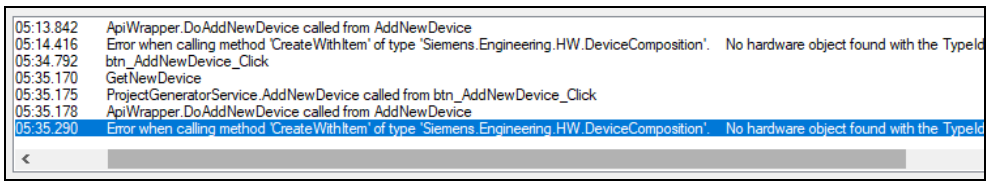


Figure 5-13

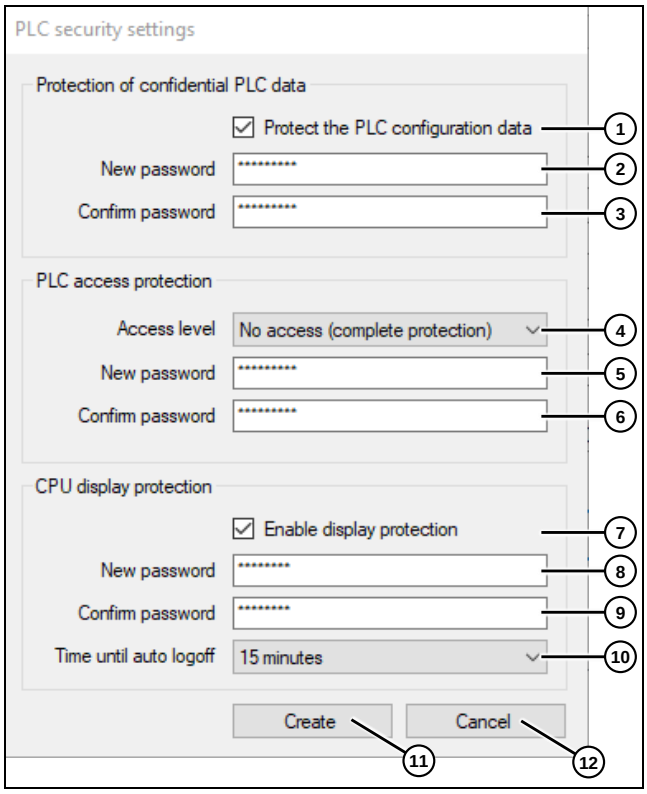


Table 5-8

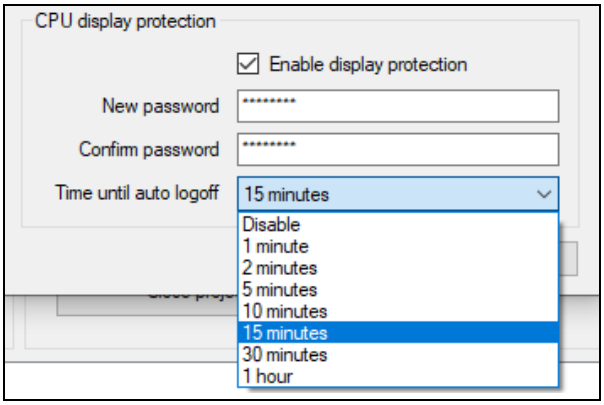
No.	Description
1.	Enable this option to secure access to PLC data with a password.
2.	The access password to the PLC configuration. The password must meet the following requirements: • At least 9 characters long • At least one capital letter • At least one number • At least one of the following special characters: @$!%*?&
3.	Repeat the password you have entered.
4.	Select the desired "Access level" (see Figure 5-11).
5.	The password for the PLC access level. The password must meet the following requirements: • At least 9 characters long • At least one capital letter • At least one number At least one of the following special characters: @$!%*?&
6.	Repeat the password you have entered.
7.	Enable this option to password-protect access via a display.
8.	The password for signing in on the display. The password must meet the following requirements: • Length between 3 and 8 characters • Only capital letters and numbers are allowed

No.	Description
9.	Repeat the password you entered.
10.	Choose the time after which the system will automatically log out on the display (see Figure 5-14).
11.	Create the security settings for the new device by clicking the "Create" button. By calling `ValidationProvider`, your input will first be validated and, if valid, the dialog will close and the security settings will be created and assigned to the new device.
12.	If you wish to abort creation of the security settings, click the "Cancel" button. Please observe the note below!

Note

If you add a new device to a project and then you abort creation of security settings, errors will be thrown when compiling this device because the security setting configuration is strictly required.

Figure 5-14



5.5 "Compile" group

Figure 5-15

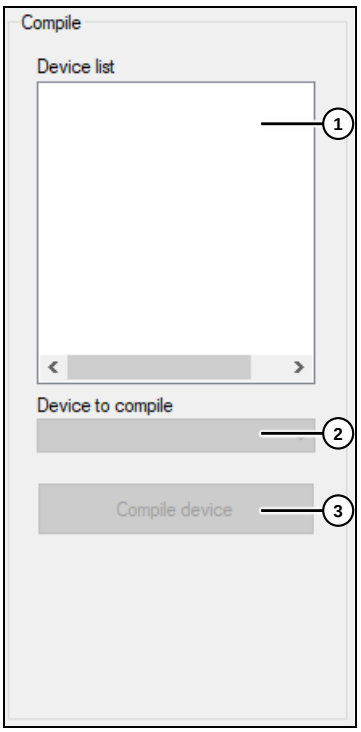


Table 5-9

No.	Description
1.	List of all devices from the open project (see also Figure 5-16).
2.	Combo Box for selecting a device to compile (see also Figure 5-16).
3.	To start compiling, click the "Compile device" button (see also Figure 5-16).

Figure 5-16

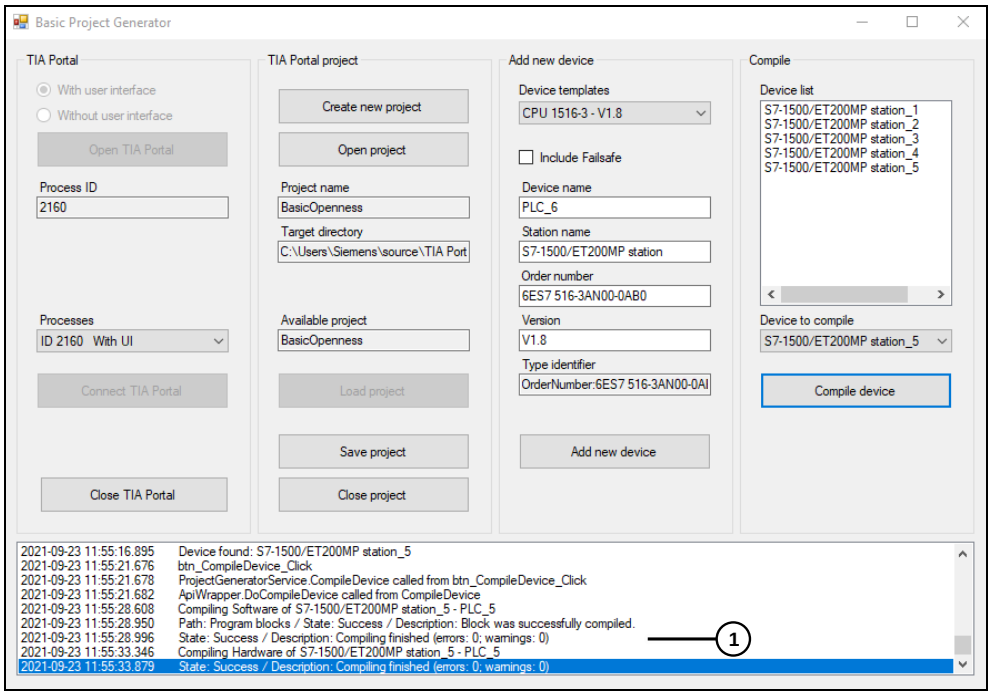
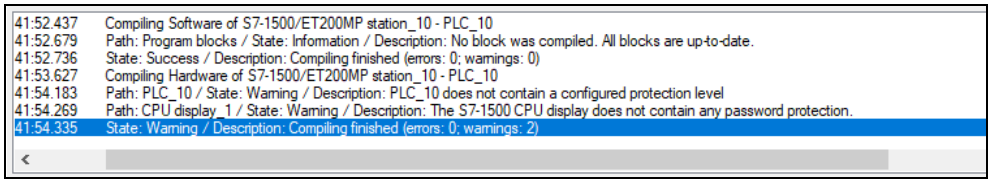


Table 5-10

No.	Description
1.	The individual compilation sub-processes are output as status information. For example, if you do not specify any "CPU display protection" in the security settings (see Figure 5-13), then a warning will be issued (see Figure 5-17).

Figure 5-17

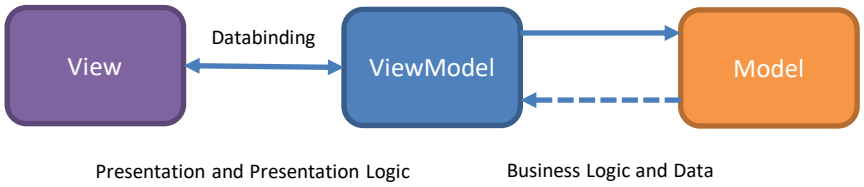


6 TIA Portal Openness Demo Application

6.1 Overview

The Model-View-ViewModel (MVVM) architectural pattern was used in developing the "TIAPortalOpennessDemo" application. See Figure 6-1 and Figure 6-8. Accordingly, the solution consists of multiple projects that are organized in various areas (see Figure 6-9). This is intended to further simplify the introduction to developing your own Openness applications. For a description of the individual areas and projects, see Table 6-4.

Figure 6-1



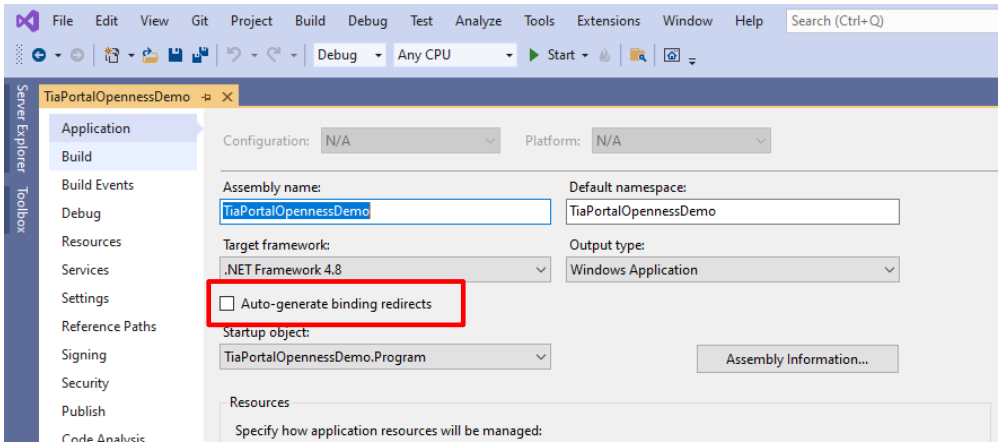
Note

To show you how to streamline development of your Openness applications for different versions, we have developed this Demo Application in such a way that you can develop and support multiple versions of the Siemens.Engineering.dll and the Openness API with only one solution and one application.

To realize this, you must reference in parallel multiple versions of the Siemens.Engineering.dll in the projects that require a reference to the Siemens.Engineering.dll. In addition, you must disable the "Auto-generate binding redirects" in the project properties in every project in this solution (see Figure 6-2, red frame).

For how to proceed in this case, see the description in Table 6-4, no. 4.

Figure 6-2



6.2 Setting up the solution

Note

This solution contains projects for the modules Step7, Sinamics Startdrive, WinCC Professional and WinCC Unified.

If you did not install the necessary software for these modules, you must unload the projects from the solution.

The same applies for the implementation of individual versions that are unavailable or which you do not intend to support. In this case as well, please unload the corresponding directories from the projects.

Everything that you unload from the solution or from a project will be retained and can be reloaded at a later time.

6.2.1 Unload project

Figure 6-3

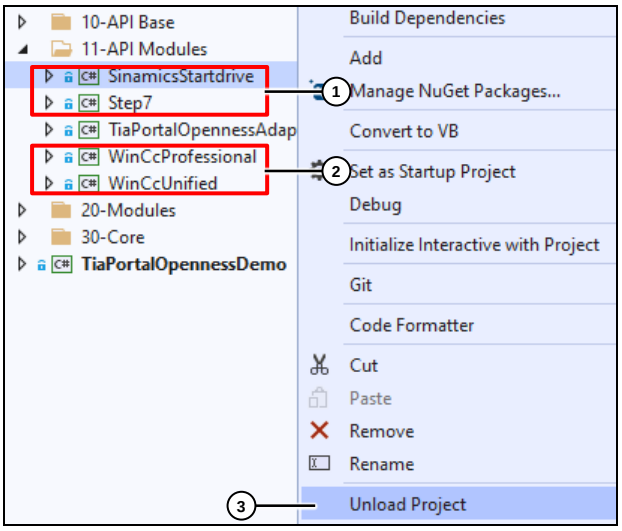


Table 6-1

No.	Description
1.	Installation-dependent projects
2.	Installation-dependent projects
3.	Unload a project from the solution using the project context menu.

6.2.2 Reload project

Figure 6-4

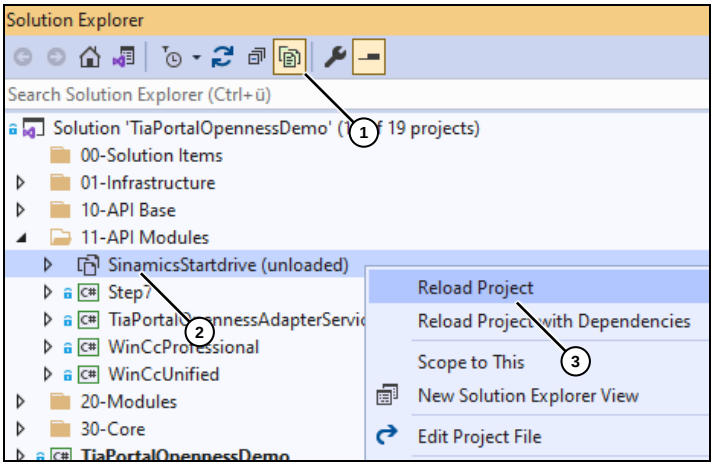
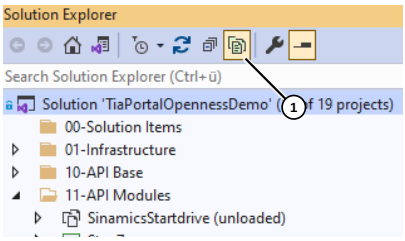


Table 6-2

No.	Description
1.	 To display the unloaded projects in the solution, the "Show All Files" function must be enabled for the solution.
2.	Select the project to be loaded.
3.	The project can be loaded to the solution with the context menu of the selected project.

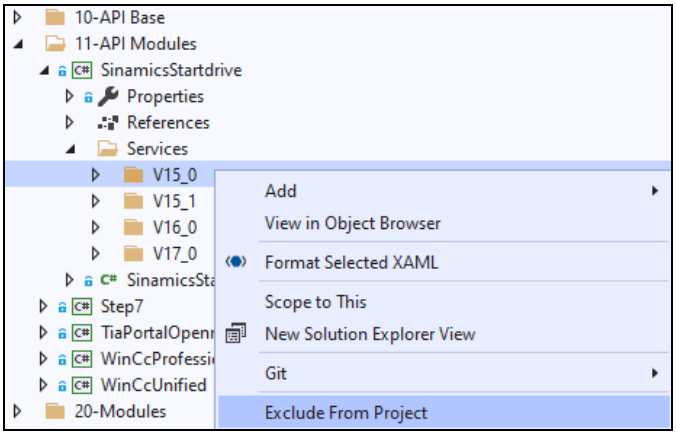
6.2.3 Exclude project folder from a project

Depending on which version you are **not** using, you must exclude the corresponding implementations for these versions from the projects. To do this, use the context menu with the function "Exclude From Project" (see Figure 6-5). This will enable you to compile the projects and/or the solution. Clean structuring of the source code is especially important for this reason.

The following projects contain version-specific implementations:

- TiaPortalOpennessAdapter – Models – V…
- SinamicsStartdrive – Services – V…
- Step7 – Services – V…
- TiaPortalOpennessAdapterService – Services – V…
- WinCcProfessional – Services – V…
- WinCcUnified – Services – V…

Figure 6-5



6.2.4 Include a project folder in a project

Figure 6-6

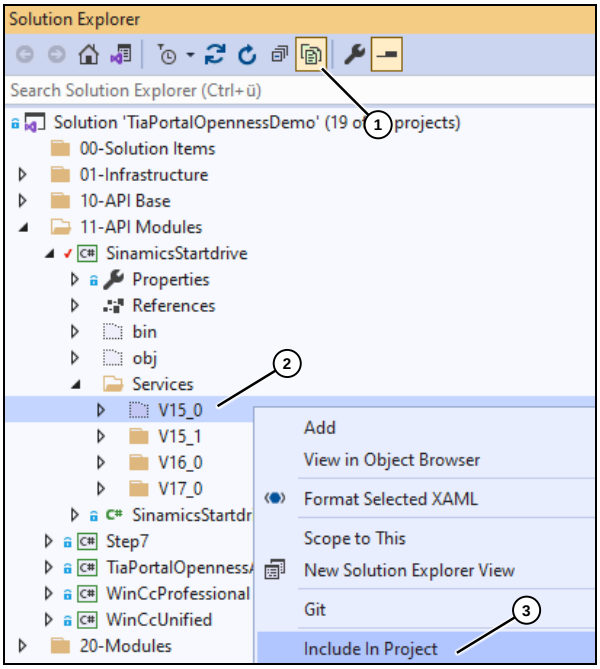


Table 6-3

No.	Description
1.	Activates the "Show All Files" function via the context menu of the project "SinamicsStartdrive".
2.	The excluded project folder that you wish to include in the project.
3.	Executes the function "Include In Project" via the context menu of the project folder.

Figure 6-7

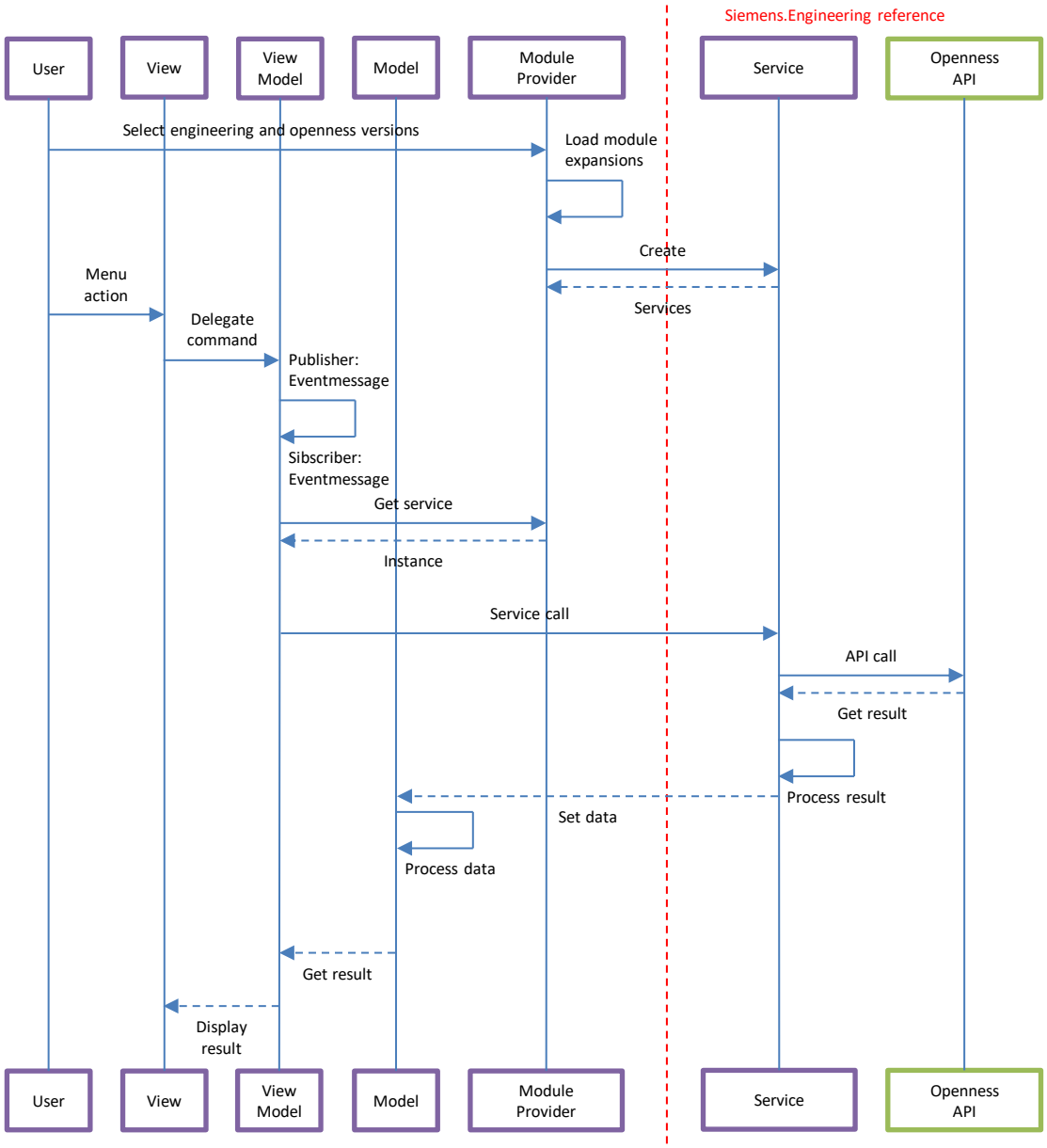


Figure 6-8

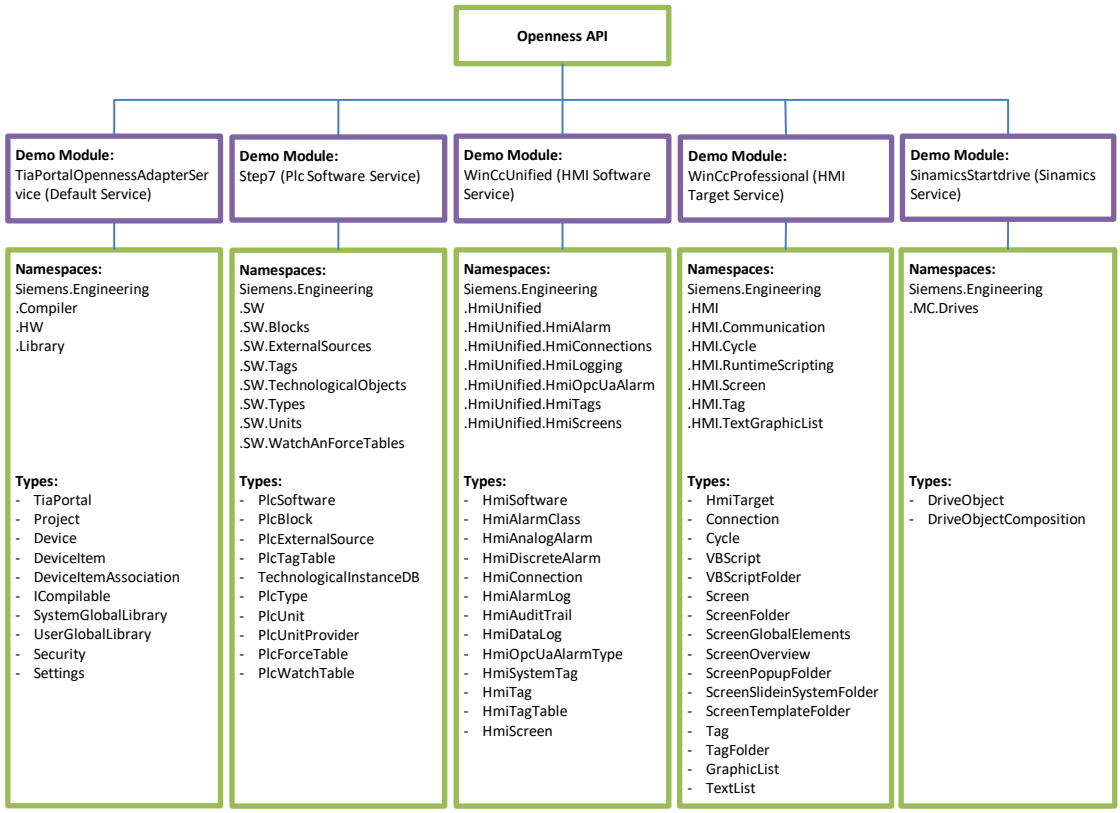


Figure 6-8 gives you an overview of the most important namespaces and types used in the TIAPortalOpennessDemo application.

To follow the description, launch the program "TIAPortalOpennessDemo.exe" and open the project with Microsoft Visual Studio.

Note

You can find a fully compiled "exe" file in the download "108716692_TIA_PortalOpenness_Demo_V17.zip" in the "TiaPortalOpennessDemo_Application.zip".

Figure 6-9

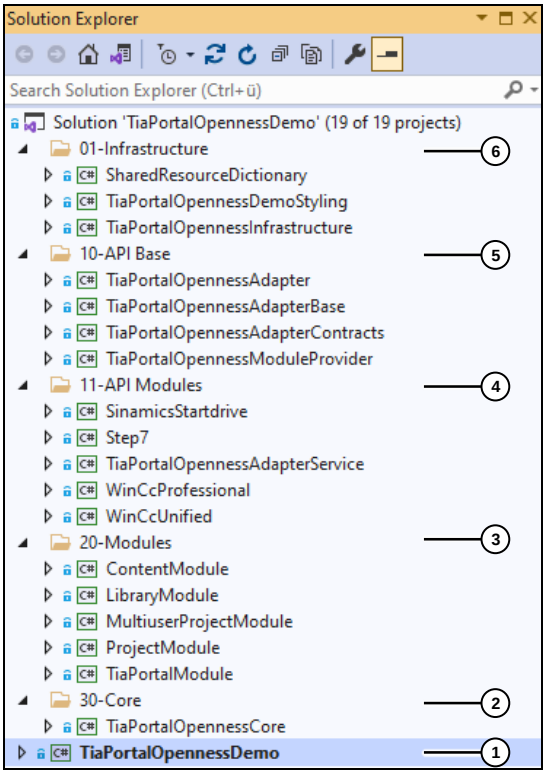
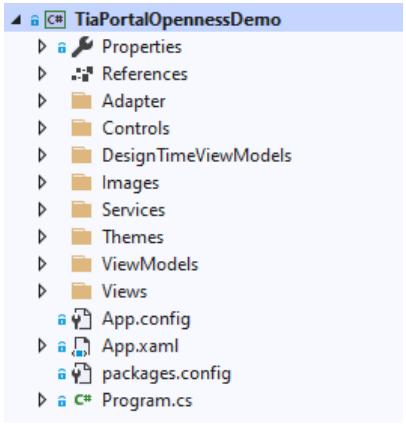
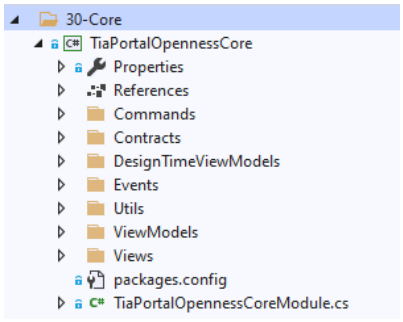
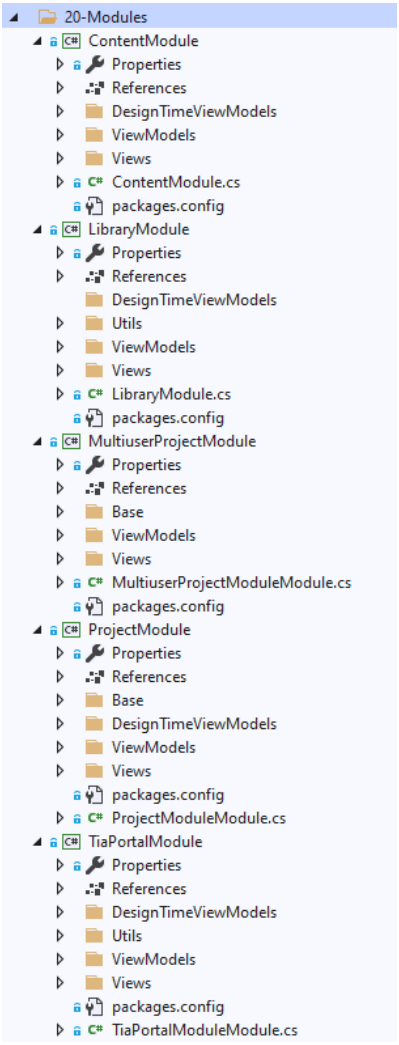
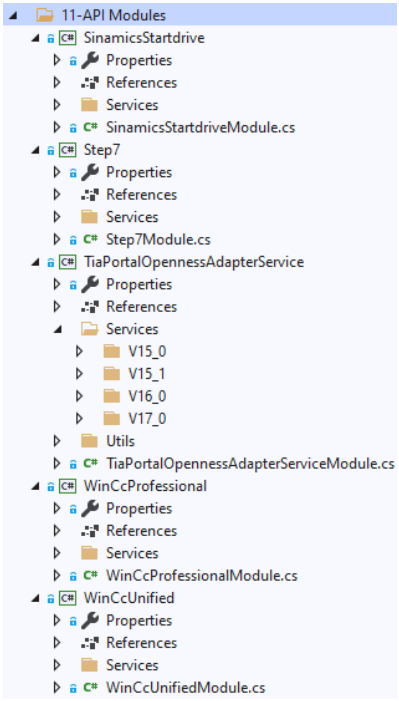


Table 6-4

No.	Description
1.	The TiaPortalOpennessDemo project is the main project in the solution.  • Adapter contains additional classes needed for MVVM in connection with the Prism framework. • Contracts contains all interface definitions for this project. • DesignTimeViewModels contains the ViewModel with sample data for the WPF Design. • Images contains the application icon. • Services contains the service implementations. • Themes contains a WPF resource. • ViewModels and Views contain the individual classes corresponding to the MVVM architectural pattern.
2.	The Core area contains the TiaPortalOpennessCore project.  • Commands contains the Command definitions. • Contracts contains all interface definitions for this project. • DesignTimeViewModels contains the ViewModel with sample data for the WPF Design. • Events contains all Event definitions, grouped by topic. • Utils contains various internal helper classes used only within this project. • ViewModels and Views contain the individual classes corresponding to the MVVM architectural pattern.

No.	Description
3.	The Modules area contains the projects for implementing the logic for individual topic areas, such as TIA Portal instance or project.  • DesignTimeViewModels contains the ViewModel with sample data for the WPF Design. • ViewModels and Views contain the individual classes corresponding to the MVVM architectural pattern. • Utils contains various internal helper classes used only within this project. • Base contains the base implementation of the ViewModel.

No.	Description
4.	The API Modules area contains the projects that provide special extensions. They can be loaded based on their use when the application starts. It should be noted that only the modules can be loaded for which the required engineering and Openness software versions are installed. The TiaPortalOpennessAdapterService is the default service and is always loaded when the application starts. It is therefore an exception.



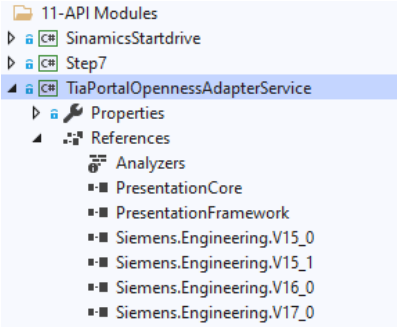
- The project structure of the individual projects is as follows:

- Services contains a substructure corresponding to the implemented version. It starts from a minimum version of V15_0. Each of these project folders may contain additional subfolders if it helps with structuring a project, and if delimitations between the version-specific implementations are required.

- The name of the subfolder, for example V17_0, is the link between the implemented API version, the engineering reference to be used via the external alias, and the parallel reference to the Siemens.Engineering.dll in multiple versions simultaneously.

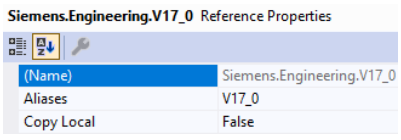
```
extern alias V17_0;
using System;
using System.Windows;
using TiaPortalOpennessAdapterBase.Base;
using TiaPortalOpennessAdapterBase.Enums;
using TiaPortalOpennessAdapterContracts;
using TiaPortalOpennessInfrastructure.Contracts;
using TiaPortalOpennessInfrastructure.Exceptions;
using V17_0::Siemens.Engineering;
using V17_0::Siemens.Engineering.HmiUnified;
using V17_0::Siemens.Engineering.HW;
using V17_0::Siemens.Engineering.HW.Features;


namespace WinCcUnified.Services.V17_0
```

- The external alias defines which reference is to be used.

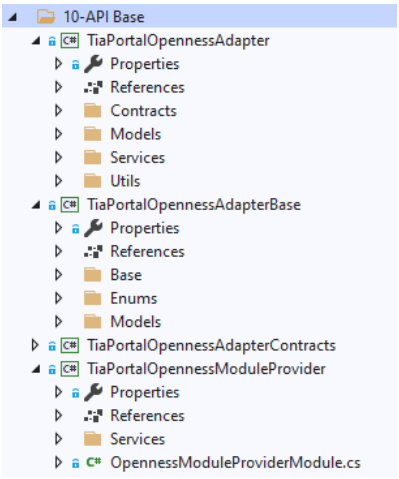
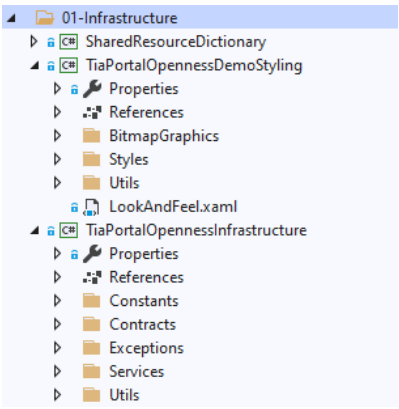


- To reference multiple versions of the Siemens.Engineering.dll, the individual DLLs must have unique file names.

- The Siemens.Engineering.dll files in version V17.0 must be renamed to Siemens.Engineering.V17_0.dll, for example.



- After you have renamed the DLL and added it as a reference, change the Properties of this reference as shown here.

No.	Description
5.	The API Base area contains the projects with the base implementations for access to the TIA Portal Openness API. Here, only the project TiaPortalOpennessAdapter has a reference to the Siemens.Engineering.dll in different versions. For how to implement this, see the description in No. 4 from this table.  • Contracts contains all interface definitions for this project. • Models contains the definition and implementation of the required data objects. • Services contains the implementation of the Service classes for the actual access operation to the TIA Portal Openness API. Direct method calls are made only in these services. • Utils contains various helper classes. • Enums contains Enum definitions as an abstract interface to the modules without reference to the Siemens.Engineering.dll. • All cross-project interfaces are defined in the project TiaPortalOpennessAdapterContracts.
6.	The project TiaPortalOpennessInfrastructure provides implementations and resources for the solution. These can be used in all projects without additional references.  • SharedResourceDictionary contains the implementation for a SharedResourceDictionary, allowing resources in the project TiaPortalOpennessDemoStyling to be managed centrally and opened in the individual projects without generating multiple instances of these resources if they are used more than once. At the same time, it defines the WPF namespace for access in xaml code. • BitmapGraphics contains all icons used in the individual projects. • Styles contains the definition of styles for various control elements. • Constants contains classes for defining constant strings. • Contracts contains all interface definitions for this project. • Exceptions contains the implementation for decoupling the API exceptions. • Services contains the service implementations. • Utils contains generic helper classes.

6.3 Assembly Resolve

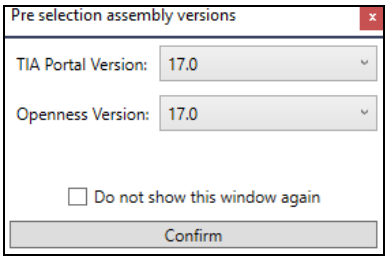
When launching the application, you must select which version of TIA Portal and TIA Portal Openness to work with (see Figure 6-10). If you only have one version in use, you can tick the box for the "Do not show this window again" option. The selected versions will be loaded automatically the next time the application starts. Confirm your selection with "Confirm".

If you ticked the box for "Do not show this window again" and wish to reset your choice at a later time so that you can select version again, this can be done through the settings (see Settings).

The installed TIA Portal versions are read from the registry using the method `GetEngineeringVersions`, which takes place in the class `Resolver` in the project `TiaPortalOpennessAdapter`, within the project folder `Utils`.

You can view the application in the project `TiaPortalOpennessDemo` in `PreSelectionAssemblyVersionViewModel` in the `GetEngineeringVersions` and `GetOpennessApiVersions` methods.

Figure 6-10



6.4 Preselection Modules

To use the functionality for Sinamics Startdrive, STEP 7, WinCC Professional or WinCC Unified, the corresponding software modules must be installed. The module expansions for the TIA Portal Openness Demo Application are then loaded from the preset directory (see Settings) and provided as a choice (see Figure 6-11).

Note

If you select the module "SinamicsStartdriveModule", it may take somewhat longer to open the project, depending on how large the project is. This is because the "SinamicsStartdriveModule" loads the drive units into the project tree. Each drive unit can have numerous parameters.

When the application is first launched, ApplicationSettings automatically selects all modules for loading. Please disable the modules whose software cannot be loaded in order to prevent possible errors when loading the modules (see Settings for further information).

Figure 6-11

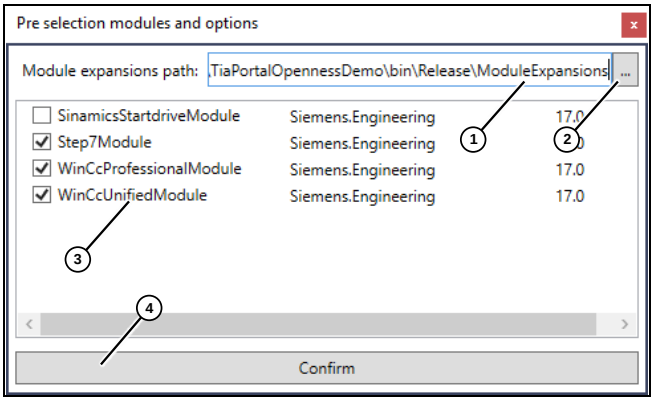


Table 6-5

No.	Description
1.	Path from which the module expansions are made available for selection, and from which the selected module expansions are loaded. This setting can be defined via the Settings.
2.	Opens a folder selection dialog to change the path if desired.
3.	List of the module expansions corresponding to the installed and loaded versions.
4.	Confirmation of the selected module expansions that will be loaded.

6.5 Main Window

The application's main view is divided into multiple panes. These display or optionally hide information about TIA Portal instances and project/global libraries in slide panels. Clicking in the content pain automatically hides the slide panels again.

Figure 6-12

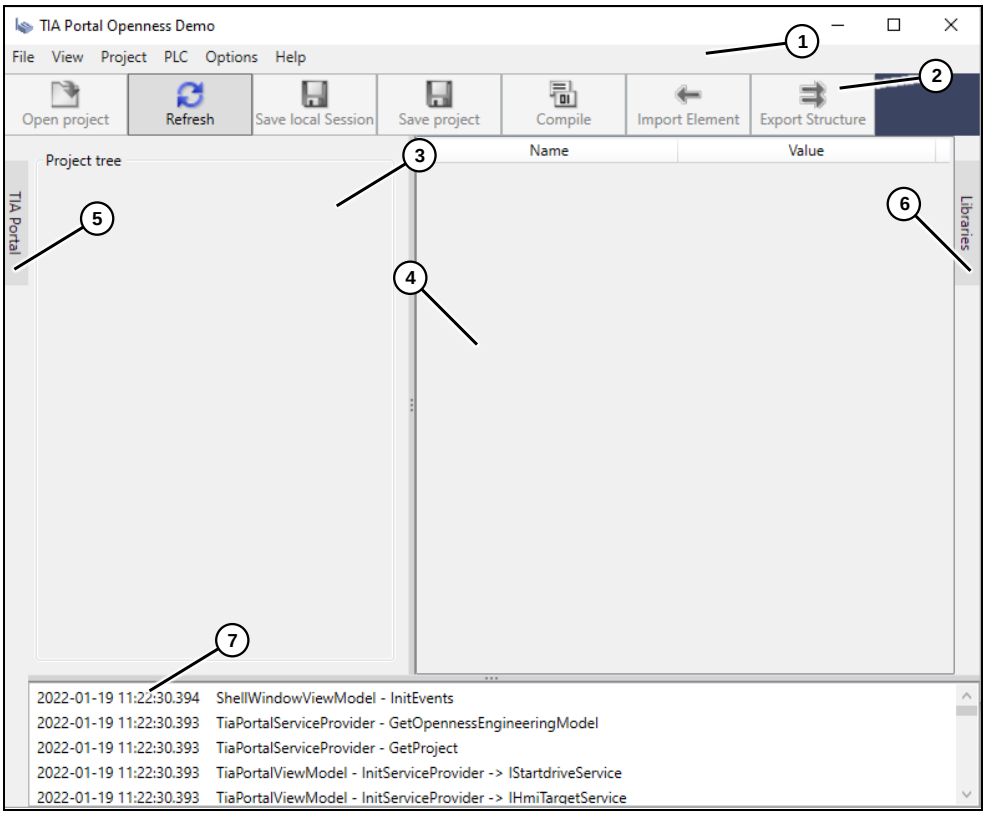


Table 6-6

No.	Description
1.	The application menu, grouped by theme.
2.	Toolbar with direct access to key functions.
3.	This pane displays the project tree. There is a distinction between a logical and a physical view. You can define which view to see via the "View" menu.
4.	This pane displays properties of an entry that has been highlighted in the project tree, along with name and property value, in list form. It uses a so-called `ContentBlackList` that defines which properties will not be displayed. This `ContentBlackList` can be found in the project `TiaPortalOpennessAdapter` in the project folder `Models.V17_0`.
5.	With this button, you can show or hide the TIA Portal slide panel (see "TIA Portal" slide panel).
6.	With this button, you can show or hide the slide panel for the project libraries and global libraries (see Slide Panel "Libraries"Slide Panel "Libraries").
7.	This pane displays information from the Trace Monitor. All activities within the process can be traced in the form of an application log. This is especially helpful when reproducing certain behaviors. The information from the Trace Monitor can also be copied to the clipboard via the About dialog (see About TIA Portal Openness Demo) and manipulated from there as the user wishes.

6.6 "TIA Portal" slide panel

Figure 6-13

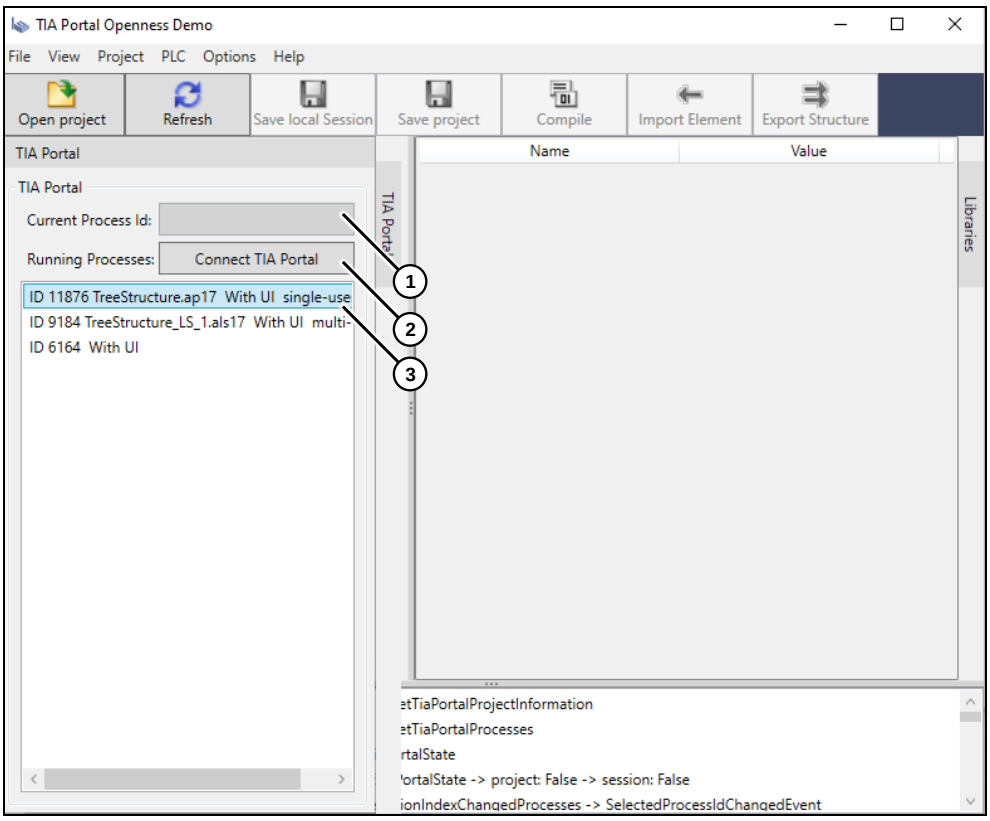


Table 6-7

No.	Description
1.	Process ID of the TIA Portal instance that the application is connected to.
2.	A connection to the TIA Portal instance highlighted in the list (see item 3) can be established via this button.
3.	List of all TIA Portal instances running on the local PC. The information about this instance is composed of the process ID, the project name, information about the start mode (with or without UI) and the type of instance (single or multi user instance).

6.7 Slide Panel "Libraries"

6.7.1 Project library

Figure 6-14

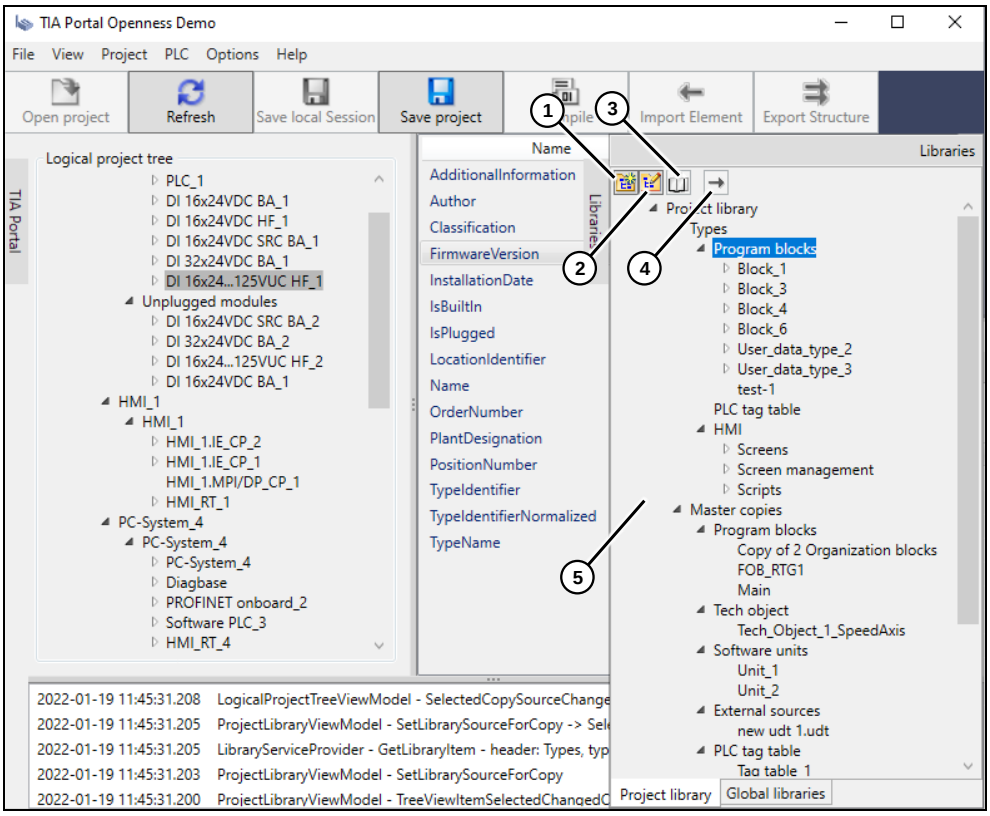


Table 6-8

No.	Description
1.	Create a new subgroup for a group highlighted in the tree (see item 5) (see Creating or editing groups).
2.	Edit the name of a group highlighted in the tree (see item 5) (see Creating or editing groups).
3.	Shows the dialog for copying a type version (see Copying a type version from the project library to the project).
4.	Export a type version highlighted in the tree (see item 5) (see Exporting type versions).
5.	The content of the project library, displayed as a tree.

6.7.1.1 Creating or editing groups

Figure 6-15

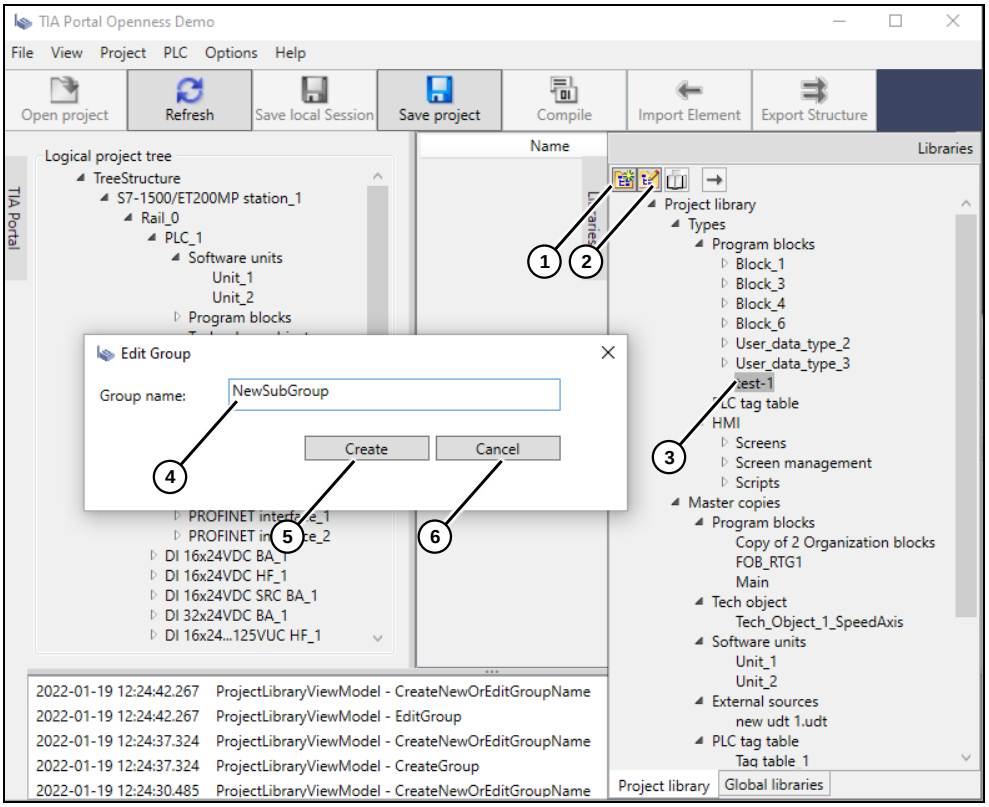


Table 6-9

No.	Description
1.	Create a new subgroup for a group highlighted in the tree.
2.	Edit the name of a group highlighted in the tree.
3.	Group highlighted in the tree.
4.	Edited name of the group highlighted in the tree.
5.	Apply changes.
6.	Cancel creation or editing.

6.7.1.2 Copying a type version from the project library to the project

Figure 6-16

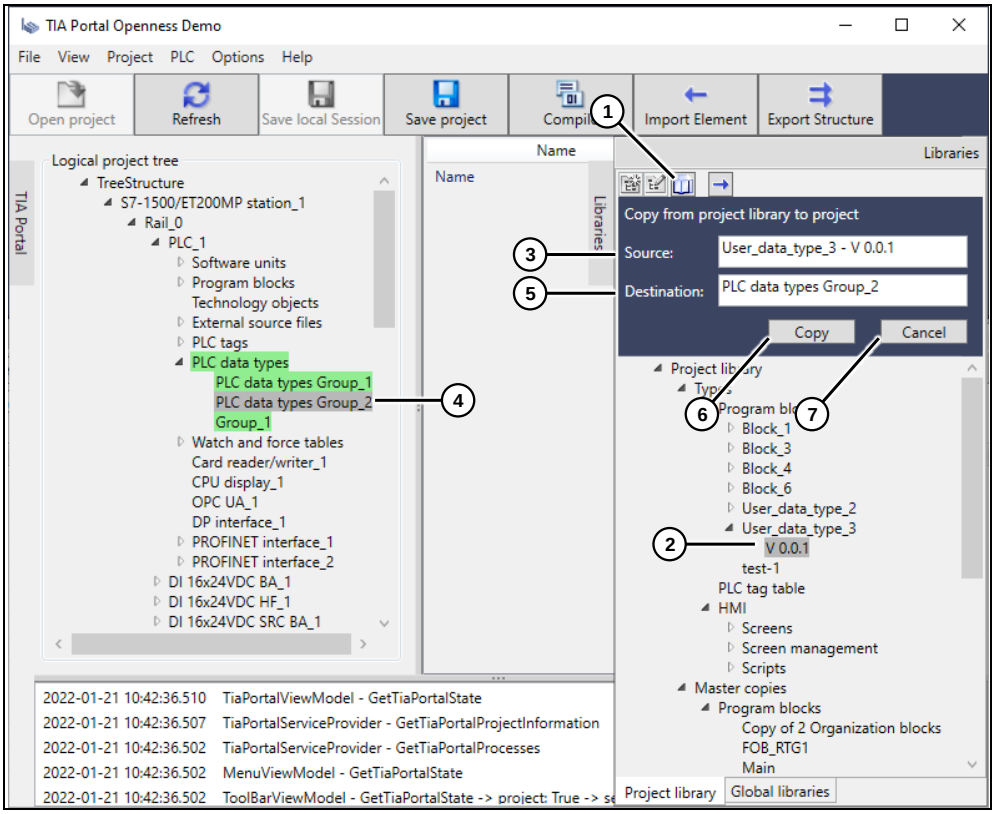


Table 6-10

No.	Description
1.	Shows the dialog for copying a type version.
2.	Type version, highlighted in the tree, to be copied. The possible copy destinations are automatically marked in green in the project tree.
3.	The type version highlighted in the tree, with name as source for the copy operation.
4.	The destination for the copy operation, highlighted in the project tree.
5.	The element highlighted in the project tree as destination for the copy operation.
6.	Start the copy operation.
7.	Cancel the copy operation.

6.7.1.3 Exporting type versions

Figure 6-17

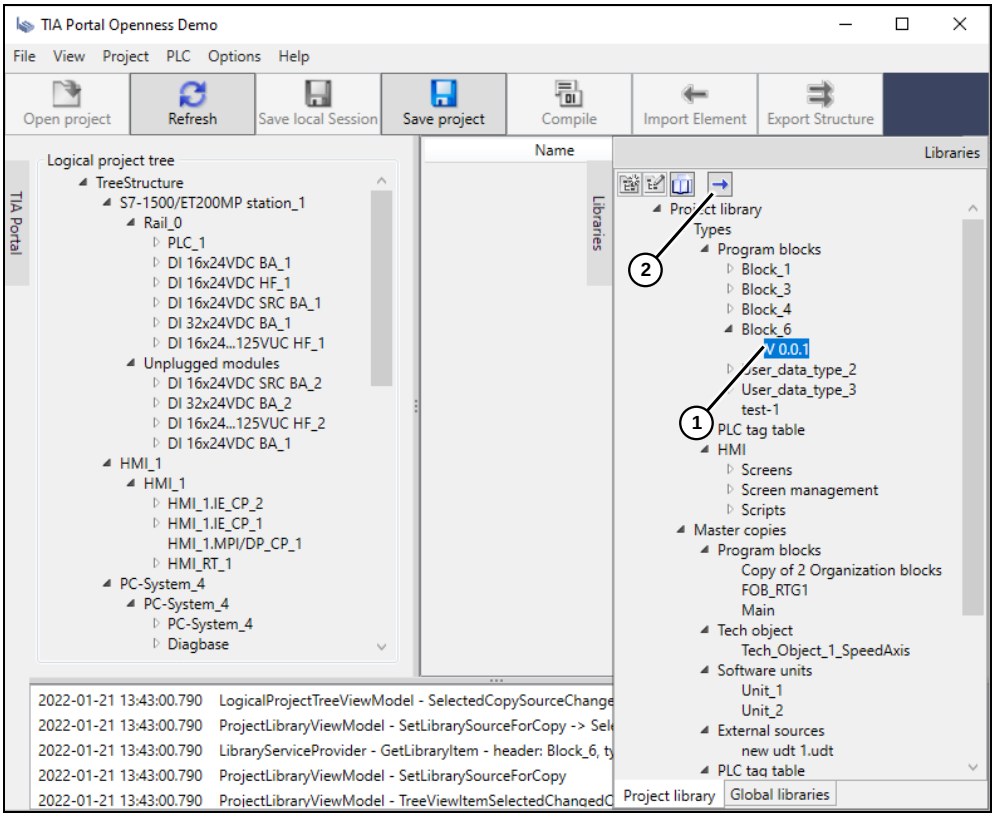


Table 6-11

No.	Description
1.	Type version, highlighted in the tree, to be exported.
2.	Starts the copy operation and generates an export file with the name of the type version in the directory set as "Export path" in the settings (see Settings).

6.7.2 Global libraries

Figure 6-18

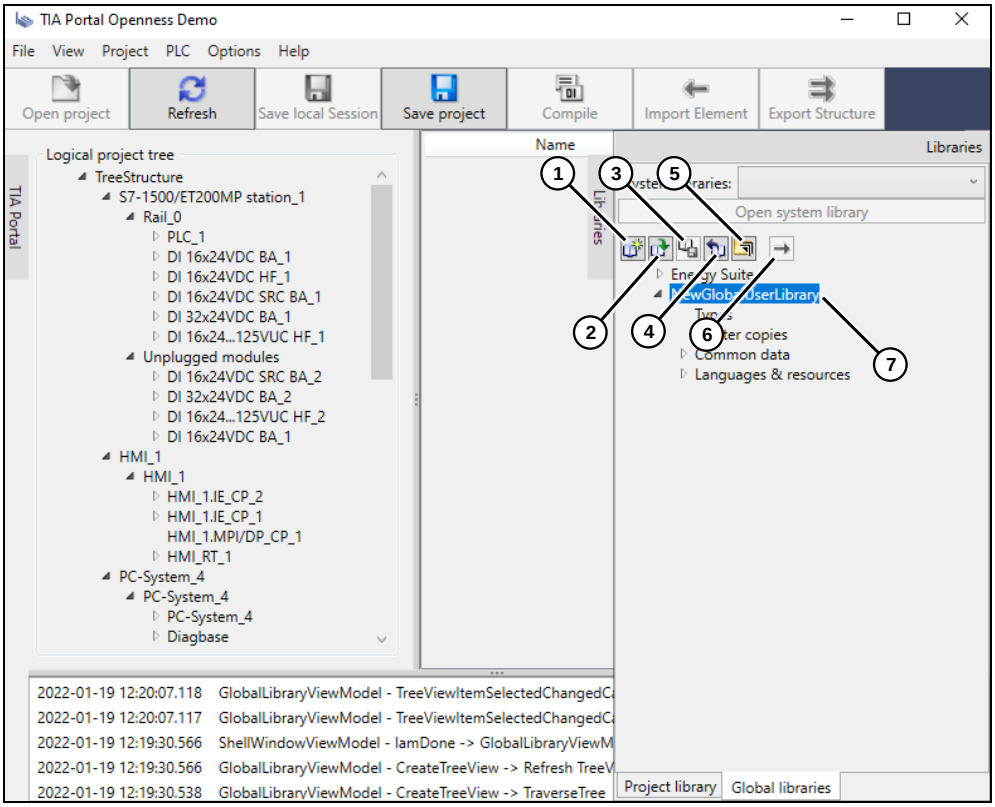


Table 6-12

No.	Description
1.	Create a new global user-defined library.
2.	Open a global user-defined library.
3.	Save changes to a global user-defined library.
4.	Close the selected global user-defined library.
5.	Update the project library with the contents of a global user-defined library.
6.	Export a type version.
7.	List of open global libraries.

Figure 6-19

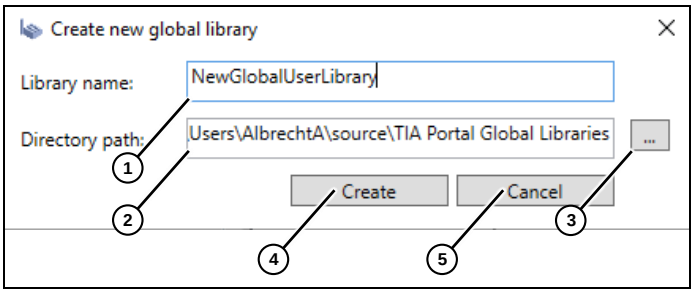


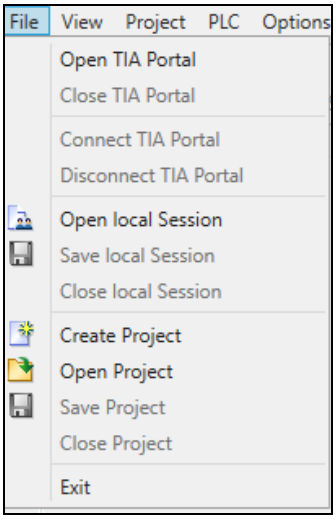
Table 6-13

No.	Description
1.	Name of the new global user-defined library to be created.
2.	Path where the new library will be saved (see Settings).
3.	Opens a folder selection dialog to change the path if desired.
4.	Create new library.
5.	Cancel action.

6.8 "File" menu

Contains actions for a TIA Portal instance, a project and the global library.

Figure 6-20



6.8.1 Open TIA Portal

You can open a new TIA Portal instance with "Open TIA Portal" in the "File" menu.

Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitServiceProvider` of class `TiaPortalViewModel`: `_tiaPortalServiceProvider = _moduleProvider.GetService(typeof(ITiaPortalServiceProvider)) as ITiaPortalServiceProvider;`

Via this instance, the method `OpenTiaPortalAsync` is called; it creates a new instance of TIA Portal and assigns the Service Property `TiaPortal` as a value.

6.8.2 Close TIA Portal

Press "Close TIA Portal" in the "File" menu to close the open TIA Portal instance, which is assigned to the Service Property `TiaPortal` as a value (see Open TIA Portal). Any project that is open will be automatically closed in the process.

The Service Instance (see Open TIA Portal) calls the method `CloseTiaPortal` which, through the Service Property `TiaPortal` (TIA Portal instance), executes the API call `TiaPortal.GetCurrentProcess().Dispose();`.

6.8.3 Connect TIA Portal

Figure 6-21

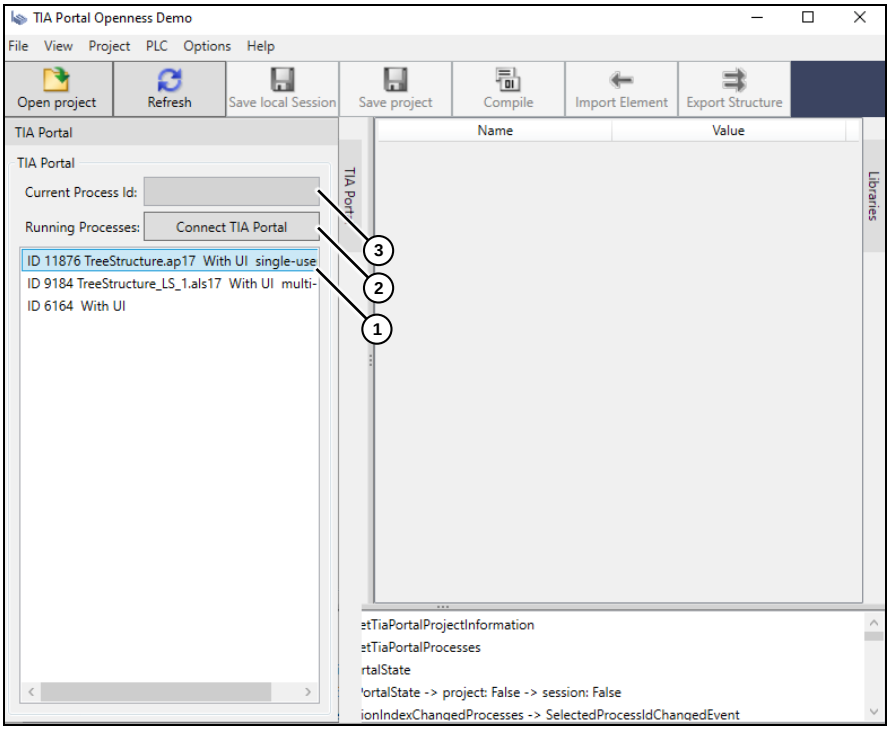
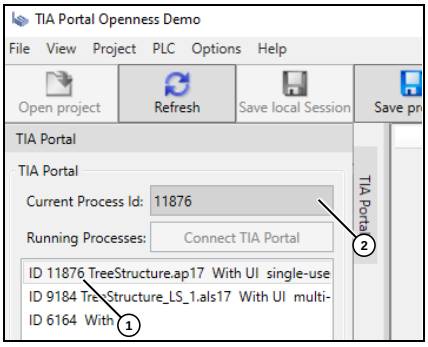


Figure 6-22



"Connect TIA Portal" in the "File" menu or the "Connect TIA Portal" button (see Figure 6-21, item 2) can establish a connection to an existing TIA Portal instance. An open project in this instance will be automatically loaded and opened for editing. To do this, select an instance from the list of all running processes (see Figure 6-21, item 1) and click the button "Connect TIA Portal" (see Figure 6-21, item 2). The Process ID (see Figure 6-22, item 1) of the instance with which the TIA Portal Openness Demo Application is connected will be shown in the "Current Process Id:" field (see Figure 6-22, item 2).

Via the Service Instance (see Open TIA Portal), a call is made to the method `ConnectTiaPortal(int processId)` which, with the API call `TiaPortal.GetProcess(processId, 5000).Attach();`, establishes a connection to the TIA Portal instance with the corresponding `processId`.

Note

Please note that the API call is not executed on the Service Property `TiaPortal`, but instead on the Siemens.Engineering object of the same name, `TiaPortal`. The returned value from `.Attach()`, then, is a TIA Portal instance which is assigned to the eponymous Service Property `TiaPortal` as a value.

6.8.4 Disconnect TIA Portal

Click "Disconnect TIA Portal" in the "File" menu to disconnect the Demo Application from an active TIA Portal instance without closing the TIA Portal instance.

The method `DisconnectTiaPortal` is called via the service instance (see Open TIA Portal). The method utilizes the API call `TiaPortal?.Dispose();` to terminate the connection to the TIA Portal instance.

Note

Please note that this API call `.Dispose()` is run on the TIA Portal instance, i.e. the Service Property `TiaPortal`.

6.8.5 Open local Session

You can use "Open local Session" in the "File" menu to open a local session instance.

Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitServiceProvider` of class `BaseMultiuserProjectViewModel`: `MultiuserServiceProvider = _moduleProvider.GetService(typeof(IMultiuserServiceProvider)) as IMultiuserServiceProvider;`

via this instance, the method `OpenLocalSessionAsync` is called in which, on the `LocalSessionComposition` of a `TiaPortal` instance, the `Open` method is called with the project file of the local session. `var localSession = tiaPortal.LocalSessions.Open(new FileInfo(projectPath));`

Note

A single-user project file has the file ending *.ap17, where 17 represents the version that this project was created with.

The project file for a local session has the file ending *.amc<Version>. A local session file, by contrast, has the file ending *.als<Version>.

This is why *.ap and *.als appear in the TIA Portal Openness Demo Application (see "TIA Portal" slide panel).

6.8.6 Save local Session

Click "Save local Session" in the "File" menu to save all changes to a local session.

The method `SaveLocalSessionAsync` is called via the service instance (see Open local Session). The method uses the API call `CurrentSession.Save();`, where `CurrentSession` is the instance of the local session.

6.8.7 Close local Session

Click "Close local Session" in the "File" menu to close a local session that has been opened.

The method `CloseLocalSessionAsync` is called via the service instance (see Open local Session); the API call

```
var localSession = TiaPortal.LocalSessions.FirstOrDefault();
localSession?.Close();
```

closes the local session.

6.8.8 Create Project

Click "Create Project" in the "File" menu to create a new project. To achieve this, you must enter the name and the destination folder where the project will be created and saved (see Figure 6-23).

Figure 6-23

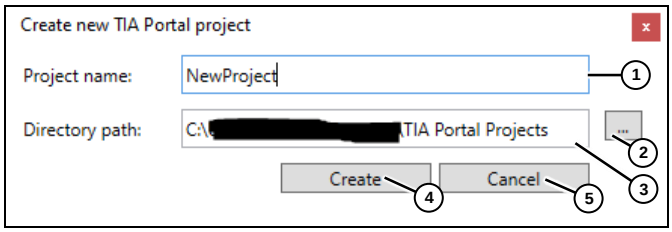


Table 6-14

No.	Description
1.	Enter a name for the new project. The name must comply with the Windows rules for file names.
2.	Open the File Explorer and select the destination directory where the new project will be created.
3.	The path where the new project will be created is read from the settings (see Settings) and displayed in the text field. It can be edited if desired.
4.	Click the "Create" button to create the new project. By calling `ValidationProvider`, your input will first be validated and, if valid, the dialog will close and the new project will be created.
5.	Click the "Cancel" button if you wish to abort the process.

Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

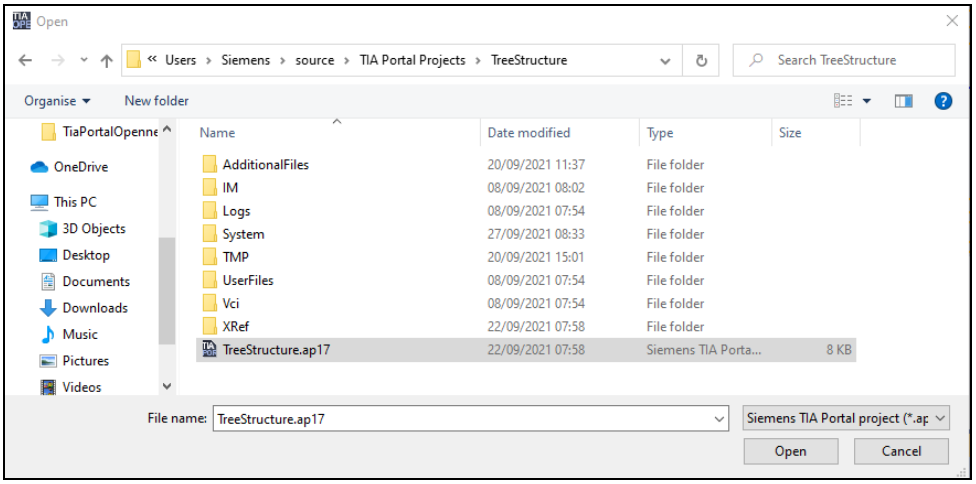
via this instance, the method `CreateProjectAsync` is called which, with the API call `var newProject = TiaPortal.Projects.Create(opennessDemoModel.ProjectModel.TargetDirectory, opennessDemoModel.ProjectModel.ProjectName);`, creates a new project.

6.8.9 Open Project

Open a project with "Open Project" from the "File" menu. To do this, use the file selection dialog to choose the desired project file (see Figure 6-24). The file filter is set to *.ap* so that all project versions will appear. The project then opens and the project data (project tree) are loaded to the application (see Figure 6-26 and Figure 6-27).

Via the service instance (see Create Project), the method `OpenProjectAsync(string projectPath)` is called. The method uses the API call `Project newProject = tiaPortal.Projects.Open(new FileInfo(projectPath));` to open the selected project, which is passed as a parameter.

Figure 6-24



6.8.10 Save Project

"Save Project" in the "File" menu saves all changes to a project.

Via the service instance (see Create Project), the method `SaveProjectAsync` is called. The method uses the API call `((Project)CurrentProject).Save();` to save all changes to a project.

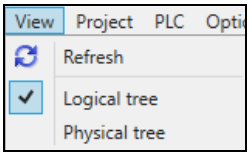
6.8.11 Close Project

"Close Project" in the "File" menu closes the open project.

Via the service instance (see Create Project), the method `CloseProjectAsync` is called. The method uses the API call `var project = TiaPortal.Projects.FirstOrDefault();` to first determine what project is open. Then the project is closed with the API call `project?.Close();`.

6.9 "View" menu

Figure 6-25



The "View" menu gives you the ability to switch between the logical and physical tree structure in the project. See Figure 6-26 and Figure 6-27.

Following the sequence diagram (see Figure 6-7), a click in the "View" menu on "Logical tree" or "Physical tree" calls the Command Handler `SetNavigationPath(string navigationPath)` in the class `MenuViewModel` in the project `TiaPortalOpennessCore`. The name of the View that the system will display/navigate to is passed as a parameter.

```
<MenuItem Header="_View">
    <MenuItem
        Command="{Binding TreeViewNavigationCommand}"
        CommandParameter="LogicalProjectTreeView"
        Header="Logical tree"
        IsChecked="{Binding ShowLogicalTree}" />
    <MenuItem
        Command="{Binding TreeViewNavigationCommand}"
        CommandParameter="PhysicalProjectTreeView"
        Header="Physical tree"
        IsChecked="{Binding ShowLogicalTree,
        Converter={StaticResource IbConverter}}" />
</MenuItem>
```

The Command Handler `SetNavigationPath(string navigationPath)` then calls the method `Navigate(string navigationPath)`, where a `_regionManager.RequestNavigate(RegionNames.ProjectTreeRegion, navigationPath);` is executed.

This causes the view selected by the menu (`CommandParameter`) to be displayed.

The ViewModel of the View finds the requisite data and prepares them for display.

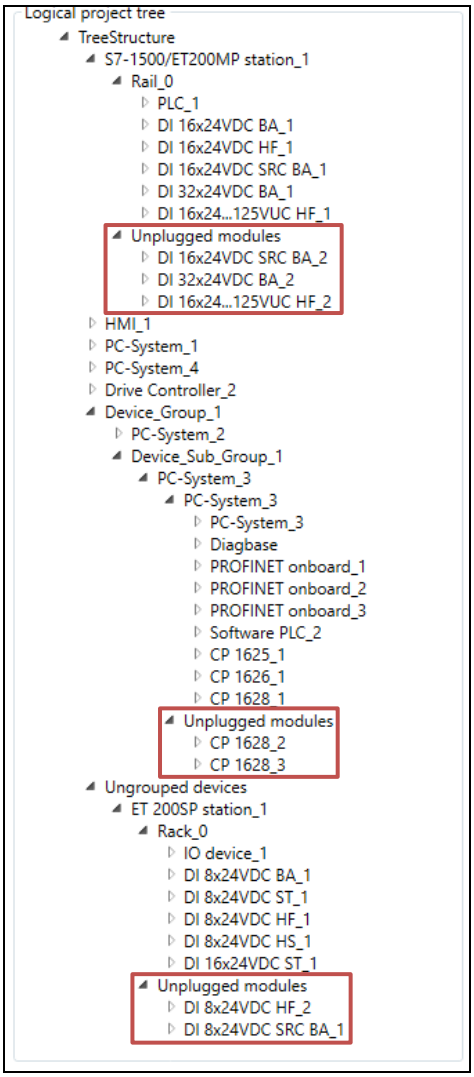
For example, to display the "Logical tree", this involves calling `_regionManager.RequestNavigate(RegionNames.ProjectTreeRegion, navigationPath);` in the class `LogicalProjectTreeViewModel` to run a `OnNavigatedTo(NavigationContext navigationContext)` in which the method `LoadTreeDataAsync(true);` is called from the base class `BaseProjectViewModel`. Then, via `ProjectService`, the method `_projectService.LoadTreeStructure(loadLogicalTree);` is called, which in turn uses multiple API calls to acquire the project tree and assign it to the Property of the Project Model `_projectModel.LogicalTree;` as a value.

6.9.1 Logical view

The logical tree structure arranges all devices under a rail or rack. All unplugged devices are arranged on the same level as a rail or rack (see red box in Figure 6-26).

One exception here is a Device Group, since it does not have a rail or rack. Therefore, the name of the device is used as an additional grouping.

Figure 6-26



6.9.2 Physical view

In the physical tree structure, all devices are arranged underneath a station; here, a rail and all devices (including the unplugged devices) are on the same level. For information on which device is unplugged and which is not, see the properties view (see Figure 6-28).

Figure 6-27

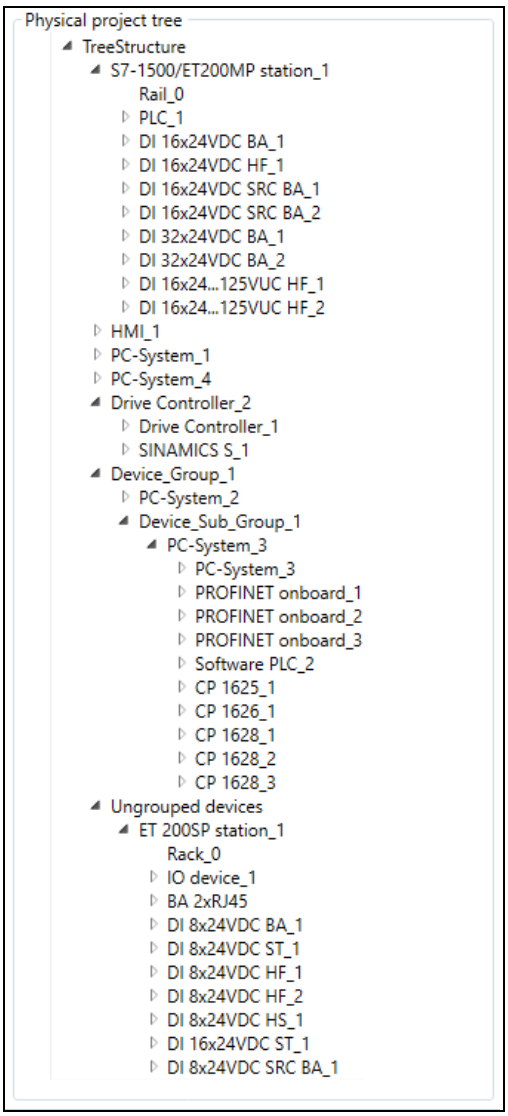
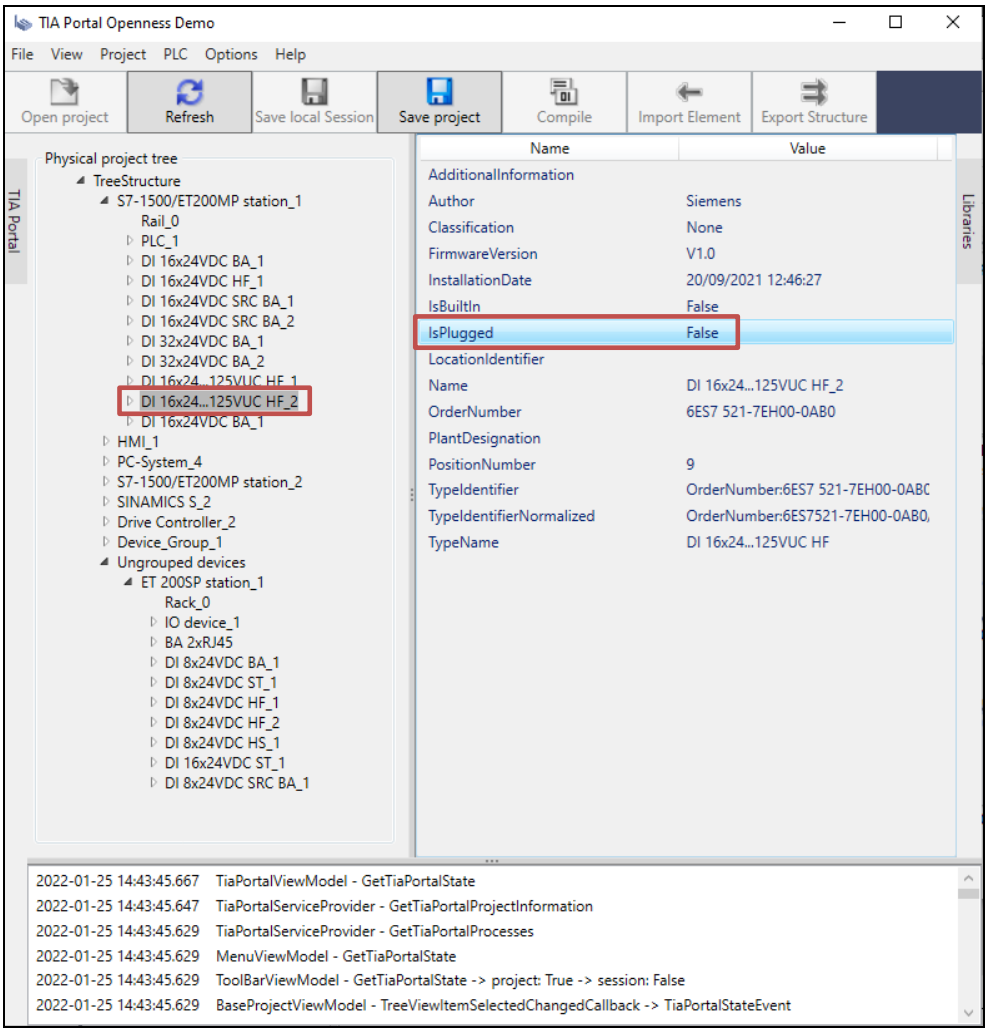


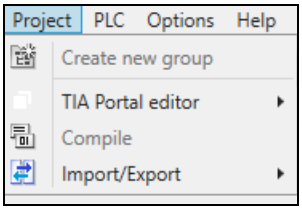
Figure 6-28



6.10 "Project" menu

In this menu you will find functions that can be run on the project level or on a project item.

Figure 6-29



6.10.1 Create new group

Figure 6-30

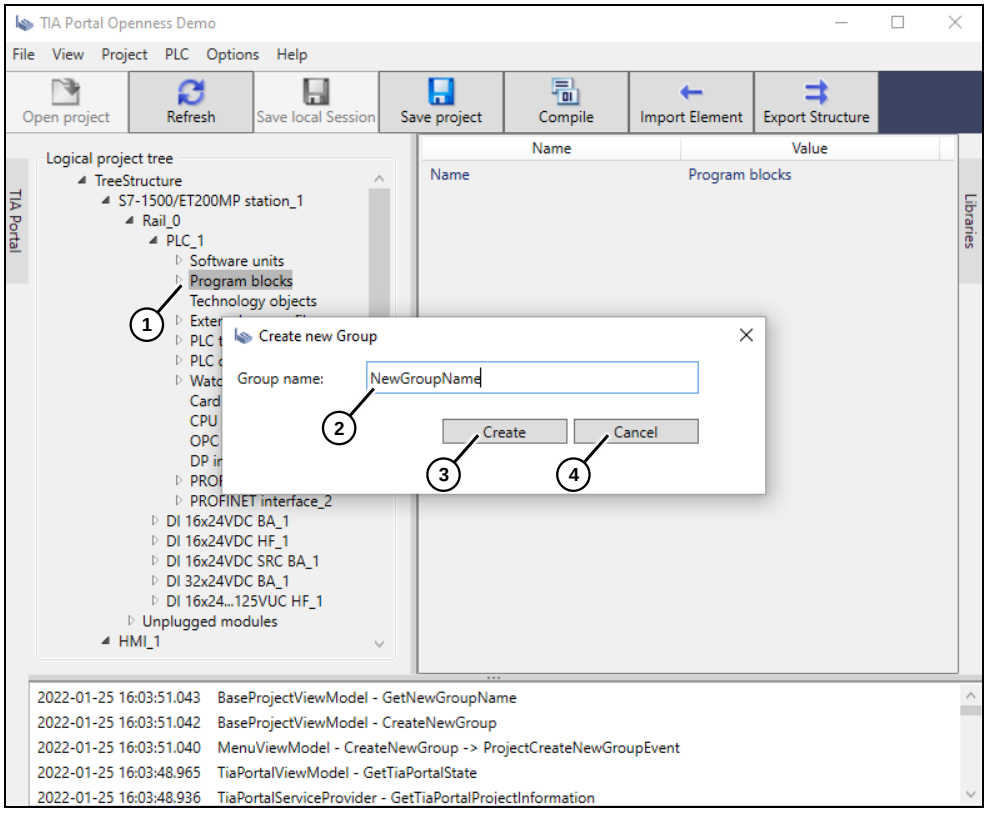


Table 6-15

No.	Description
1.	Highlighted element in the project tree for which a new subgroup will be created.
2.	Name for the new group.
3.	Create new group (as subgroup) for the highlighted element and close the dialog.
4.	Cancel operation and close dialog.

Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

The call `ProjectServiceProvider.CreateNewGroup(newGroupName, (string)SelectedItem.Header, (Guid)SelectedItem.Tag, true);` is made via this instance. Here, the name of the group being created, the name of the highlighted element (see Figure 6-30, item 1) and the GUID of the highlighted element are passed as parameters. The fourth parameter in the call indicates that we are working on the logical project tree and that the project item should be found within it.

The project item is found with this information and the `var parentProjectItem = GetProjectItem(parentGroup, tag, logicalTree);` `call`.

The project item found is passed both to `GroupEditorService`, a subservice of `ProjectServiceProvider`, as well as to an instance of `_plcSoftwareService`. Each service decides for itself whether it can process the project item or not.

```
if (parentProjectItem != null)
{
  using (var groupService = new  roupEditorService(_traceLogService))
  {
    groupService.CreateGroup(parentProjectItem, groupName);
  }
  _plcSoftwareService?.CreateGroup(parentProjectItem.DeviceItem,
                                   groupName);
}
```

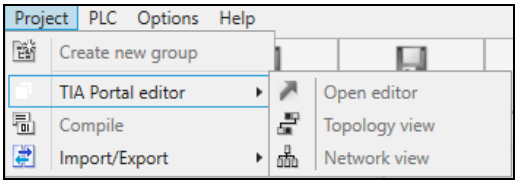
If the highlighted element is a `DeviceUserGroup`, then a new group is created for the item via `groupService` with the API call `typeGroup.Groups.Create(groupName);` and the `DeviceUserGroupComposition` is added.

If, on the other hand, the highlighted element is a PLC type, then `_plcSoftwareService` is used to determine what kind of PLC type it actually is; with an API call to its type-specific -`UserGroupComposition.Groups.Create(groupName);`m a new group is created and added.

6.10.2 "TIA Portal editor" submenu

Via this menu, you can control the view in TIA Portal and display a highlighted project item in TIA Portal. In reality, this makes it a kind of remote control.

Figure 6-31



6.10.2.1 Open Editor

"Open editor" in the menu "Project -> TIA Portal editor" lets you open the corresponding hardware editor in TIA Portal for a highlighted element in the project tree, then edit the project item within.

Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

Depending on the view selected, the method `OpenEditor` is called in either the class LogicalProjectTreeViewModel or PhysicalProjectTreeViewModel. In that method, the `OpenEditor` method of the same name is called on the service instance.

Based on the name (header) and the GUID, the service determines the associated project item and executes the actual API call. If the project item is not of type Device, an attempt is made by reflection to find the method "`ShowInEditor`". If the project provides such a method, it will be run. Otherwise, the project item cannot be displayed in any hardware editor.

```
var projectItem = GetProjectItem(header, tag, logical);
if (projectItem != null)
{
  if (projectItem.DeviceItem is Device projectItemDevice)
  {
    projectItemDevice.ShowInEditor(View.Device);
  }
  else
  {
    var type = projectItem.DeviceItem.GetType();
    var methodInfo = type.GetMethod("ShowInEditor");
    if (methodInfo == null && type.BaseType != null)
    {
      methodInfo = type.BaseType.GetMethod("ShowInEditor");
    }
    if (methodInfo != null)
    {
      methodInfo.Invoke(projectItem.DeviceItem, null);
    }
  }
}
```

6.10.2.2 Topology view

"Topology view" in the menu "Project -> TIA Portal editor" lets you open the Topology view for the project in TIA Portal (see Figure 6-32).

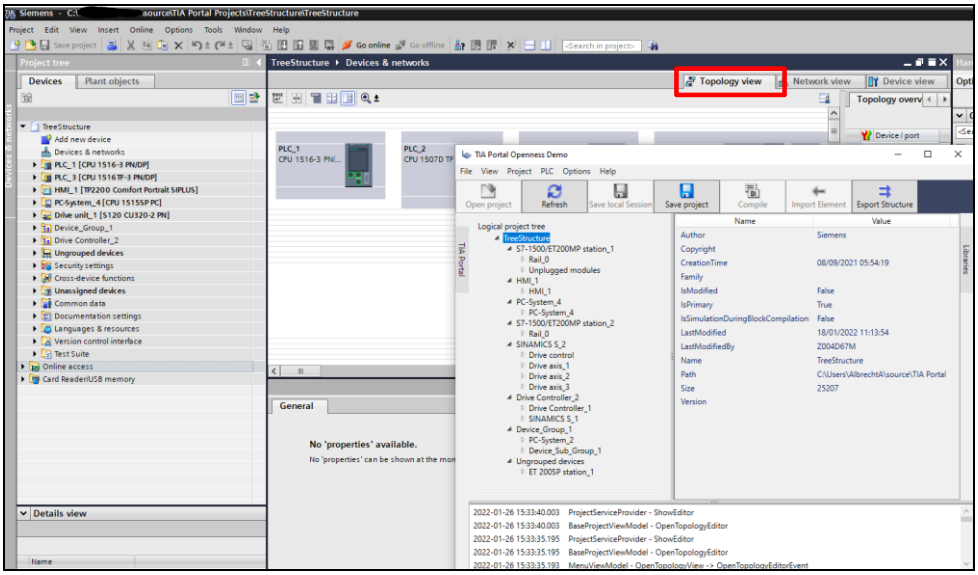
Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

In the class `BaseProjectViewModel`, the method `OpenTopologyEditor` is called in which, on the service instance, the call `ProjectServiceProvider.ShowEditor("Topology");` is run.

The service first uses the view named "Topology" to determine the corresponding View Type and then executes the API call.

```
var viewType = EnumService.GetEnumValue<View>(viewName);
CurrentProject.ShowHwEditor(viewType);
```

Figure 6-32



6.10.2.3 Network view

"Network view" in the menu "Project -> TIA Portal editor" lets you open the Network view for the project in TIA Portal (see Figure 6-33).

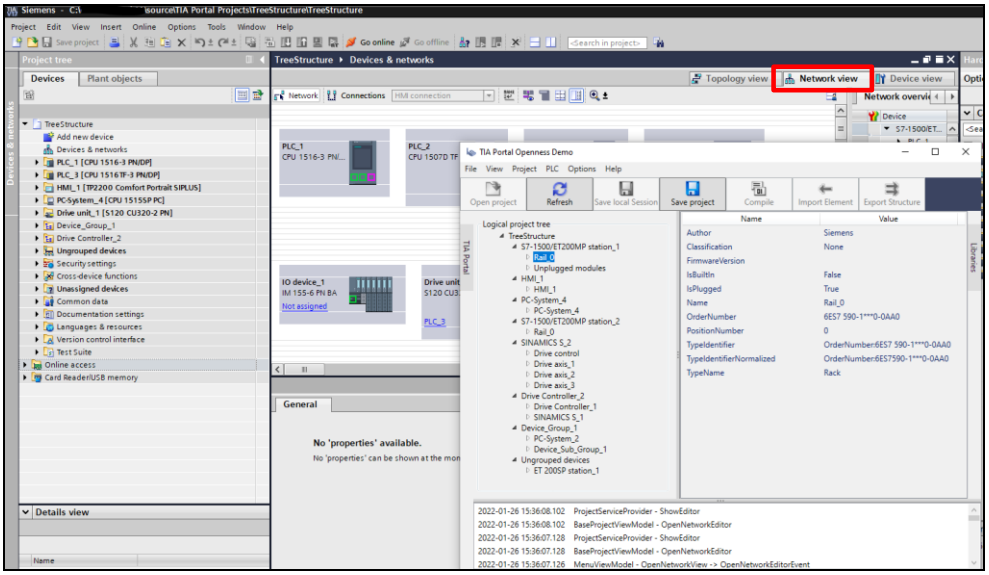
Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

In the class `BaseProjectViewModel`, the method `OpenNetworkEditor` is called in which, on the service instance, the call `ProjectServiceProvider.ShowEditor("Network");` is executed.

The service first uses the view named "Network" to determine the corresponding View Type and then executes the API call.

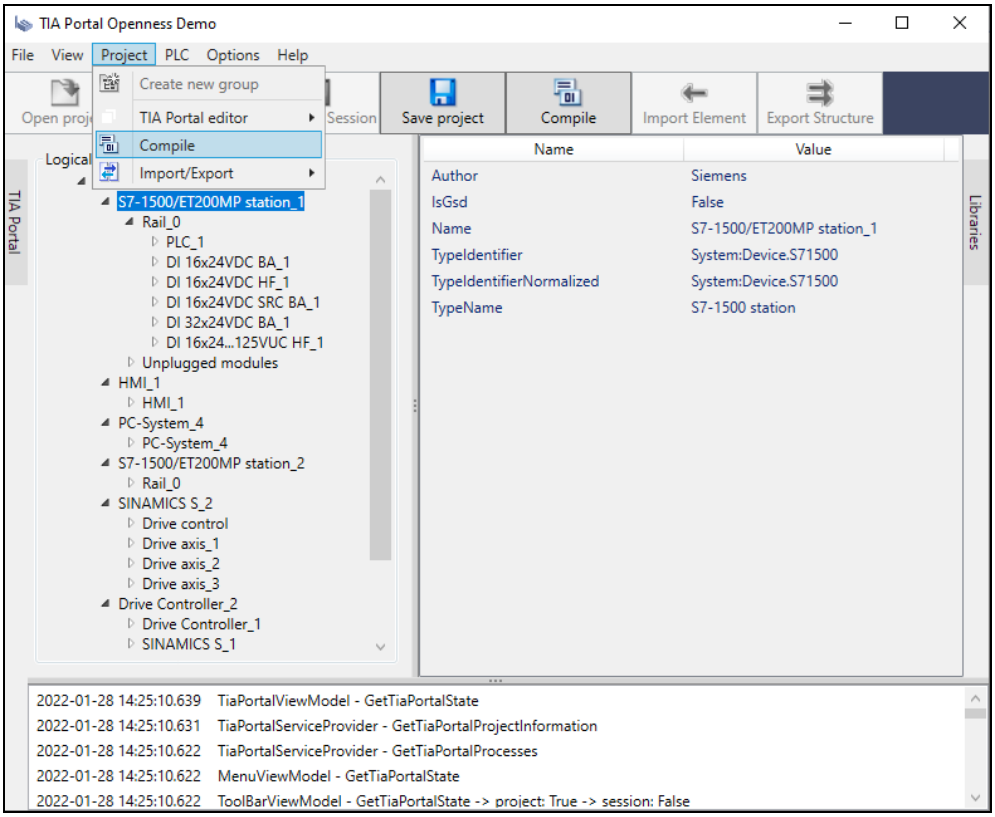
```
var viewType = EnumService.GetEnumValue<View>(viewName);
CurrentProject.ShowHwEditor(viewType);
```

Figure 6-33



6.10.3 Compile

Figure 6-34



Compiles the highlighted element in the project tree so long as the object implements the interface ICompilable (see Figure 6-34).

Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

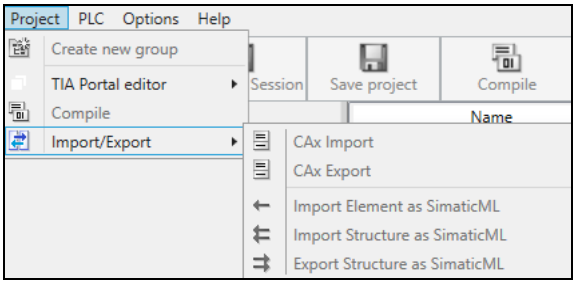
Depending on the view selected, the method `DoCompile` is called in either the class LogicalProjectTreeViewModel or PhysicalProjectTreeViewModel. In that method, the method of the same name is called on the service instance.
`ProjectServiceProvider.DoCompile((string)SelectedItem.Header, (Guid)SelectedItem.Tag, navigationContext.ShowLogicalTree);`

The project item and the compiler are determined for the actual API call. Here, the compiler is given the project item to be compiled. The `_compilerResult` is evaluated and written to the Trace output.

```
var projectItem = GetProjectItem(header, tag, logical);
if (projectItem != null)
{
  var methodInfo = GetGenericMethodInfo(projectItem,
                                         typeof(ICompilable));
  if (methodInfo != null)
  {
    var compiler = GetCompiler(methodInfo, projectItem);
    if (compiler != null)
    {
      _compilerResult = compiler.Compile();
      var compilerMessage = "Compiling " + projectItem.Header;
      WriteTraceLogProxy(processInfo.ProcessMessage + " - " +
                                                  compilerMessage);
      if (_compilerResult.Messages.Count > 0)
      {
        if (_compilerResult.Messages != null &&
                                  _compilerResult.Messages.Count > 0)
        {
          GetCompilerMessages(string.Empty, string.Empty,
                                          _compilerResult.Messages);
        }
      }
    }
  }
}
```

6.10.4 "Import/Export" submenu

Figure 6-35



6.10.4.1 CAx Import

CAx Import is used to import device data in AML format. The following import options are supported for this.

Figure 6-36

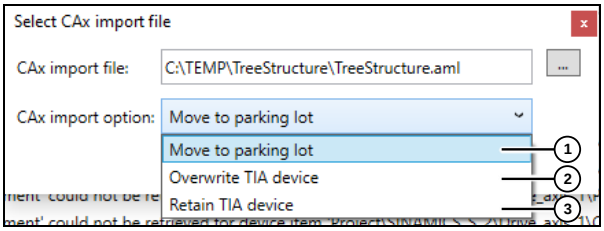


Table 6-16

No.	Description
1.	If name conflicts occur when importing the CAx data, the CAx data for the devices with a name conflict are placed in a placeholder folder.
2.	If name conflicts occur when importing the CAx data, the data for devices with a name conflict are overwritten in the TIA Portal project with the imported CAx data.
3.	If name conflicts occur when importing the CAx data, the CAx data with name conflicts are ignored and not imported.

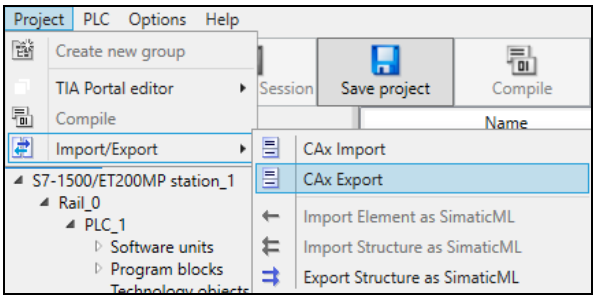
```
if (CurrentProject != null)
{
  using (var importService = new ImportService(_traceLogService))
  {
    importService.CaxImport((Project)CurrentProject, importFileInfo,
                                                      importOption);
  }
  return true;
}
```

For the actual API call, first the import provider

`var importProvider = project.GetService<CaxProvider>();` is found, on which the API call `importProvider.Import(caxImportFileInfo, logFileInfo, caxImportOption);` is then made.

6.10.4.2 CAx Export

Figure 6-37



Device data are exported in AML format with the CAx export. The CAx export is possible on the project level or on the device level.

```
if (CurrentProject != null)
{
  using (var exportService = new ExportService(_traceLogService,
                                                _settingsService))
  {
    exportService.CaxExport((Project)CurrentProject);
  }
  return true;
}
```

For the actual API call, first the import provider

`var exportProvider = project.GetService<CaxProvider>();` is found, on which the API call `exportProvider.Export(project, caxExportFileInfo, logFileInfo);` is then made.

6.10.4.3 Import Element as Simatic ML

Figure 6-38:

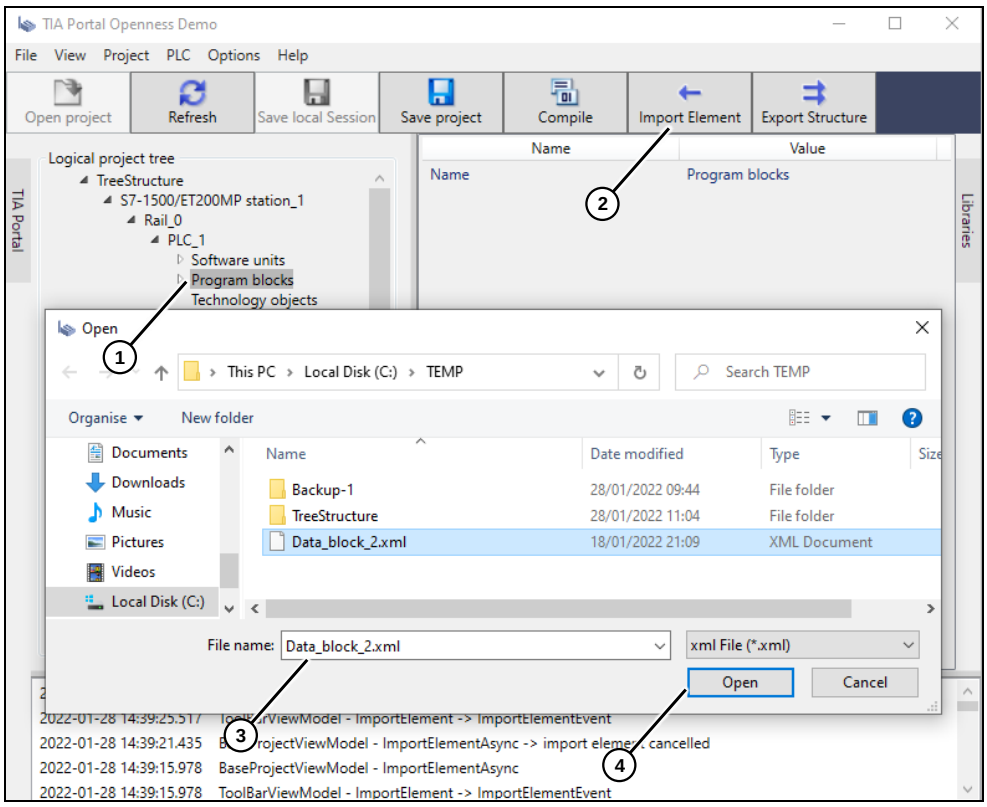
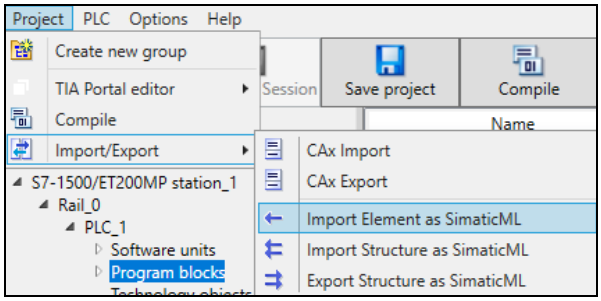


Table 6-17

No.	Description
1.	The highlighted area where an element will be imported to.
2.	Run the "Import Element" function via the toolbar or via the menu "Project > Import/Export > Import Element as SimaticML" (see Figure 6-39:).
3.	Select the XML data to be imported.
4.	Confirm the import data and call the function.

Figure 6-39:



Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`:

```
ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;
```

Depending on the view selected, the method `ImportElementAsync` is called in either the class LogicalProjectTreeViewModel or PhysicalProjectTreeViewModel. In that method, the method of the same name is called on the service instance.

```
ProjectServiceProvider.ImportElementAsync(fileInfos, true, (string)SelectedItem.Header, (Guid)SelectedItem.Tag, LogicalTreeView);
```

6.10.4.4 Import Structure as Simatic ML

Figure 6-40:

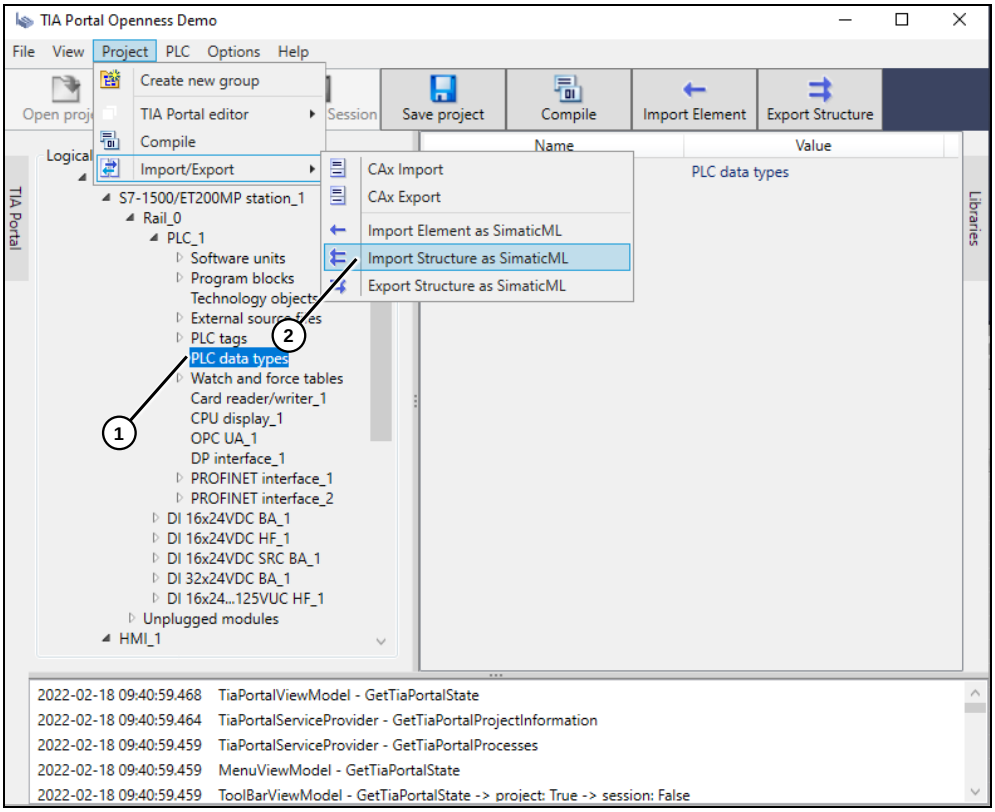


Table 6-18

No.	Description
1.	Highlighted area where a subordinate structure will be imported to.
2.	Starting the function "Import Structure as SimaticML" first opens a selection dialog where you can choose the folder that you want to import as a subordinate structure (see Figure 6-41). All subfolders and the types therein will be imported along with.

Note

Please note that the selected import directory itself (see Figure 6-41), as the root directory, is not imported. Only all subfolders in the selected directory will be imported.

Figure 6-41

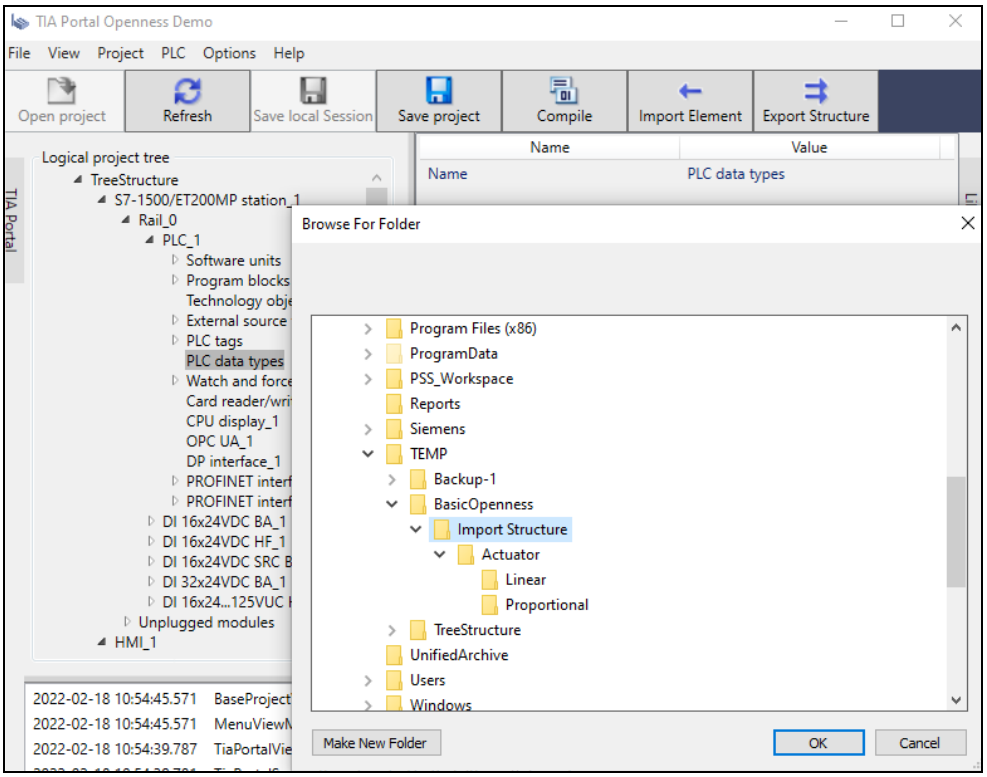
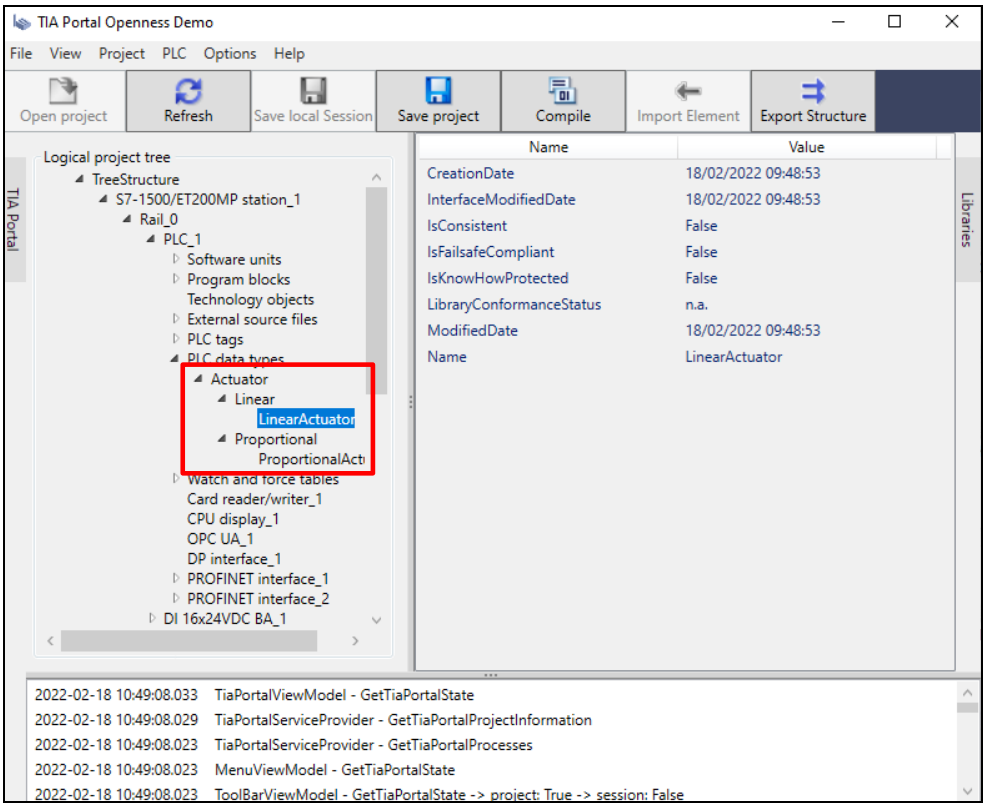


Figure 6-42



Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

Depending on the selected view, the method `ImportStructureAsync` is called via the class `LogicalProjectTreeViewModel` or `PhysicalProjectTreeViewModel` in the base class `BaseProjectViewModel`. In that method, the method of the same name is called on the service instance.
`ProjectServiceProvider.ImportElementAsync(folderBrowser.SelectedPath, (string)SelectedItem.Header, (Guid)SelectedItem.Tag, LogicalTreeView);`

Running the API functions for importing the structure requires an `ExclusiveAccess`. A check is run for which type of structure will be imported – `CheckIsPlcStructureDestination` or `CheckIsHmiTargetStructureDestination` – and the corresponding import service is run for this type (`_plcSoftwareService?.ImportStructure` or `_hmiTargetService?.ImportStructure`), as long as the required module was loaded when the application started.

```
using (var exclusiveAccess = tiaPortal?.ExclusiveAccess("Import
element"))
  {
    if (exclusiveAccess != null)
    {
      if (CheckIsPlcStructureDestination(projectItem.
                                          DeviceItem))
      {
        using (var transaction =
                        exclusiveAccess.Transaction(CurrentProject,
                        "Import structure"))
        {
          bool? result = _plcSoftwareService?.ImportStructure
                                            (projectItem.DeviceItem,
                                             importFolderPath,
                                             ImportOptions.Override);
          transaction.CommitOnDispose();
          if (result != null && result == true)
          {
            WriteTraceLogProxy(processInfo.ProcessMessage + " - " +
            nameof(_plcSoftwareService) + " - Import structure
            finished succeed");
          }
        }
      }
      if (CheckIsHmiTargetStructureDestination(projectItem.
                                                DeviceItem))
      {
        using (var transaction =
                        exclusiveAccess.Transaction(CurrentProject,
                        "Import structure"))
        {
          bool? result = _hmiTargetService?.ImportStructure
                                           (projectItem.DeviceItem,
                                            importFolderPath,
                                            ImportOptions.Override);
          transaction.CommitOnDispose();
          if (result != null && result == true)
          {
            WriteTraceLogProxy(processInfo.ProcessMessage + " - " +
            nameof(_hmiTargetService) + " - Import structure
            finished succeed");
          }
        }
      }
    }
  }
```

6.10.4.5 Export Structure as Simatic ML

Figure 6-43:

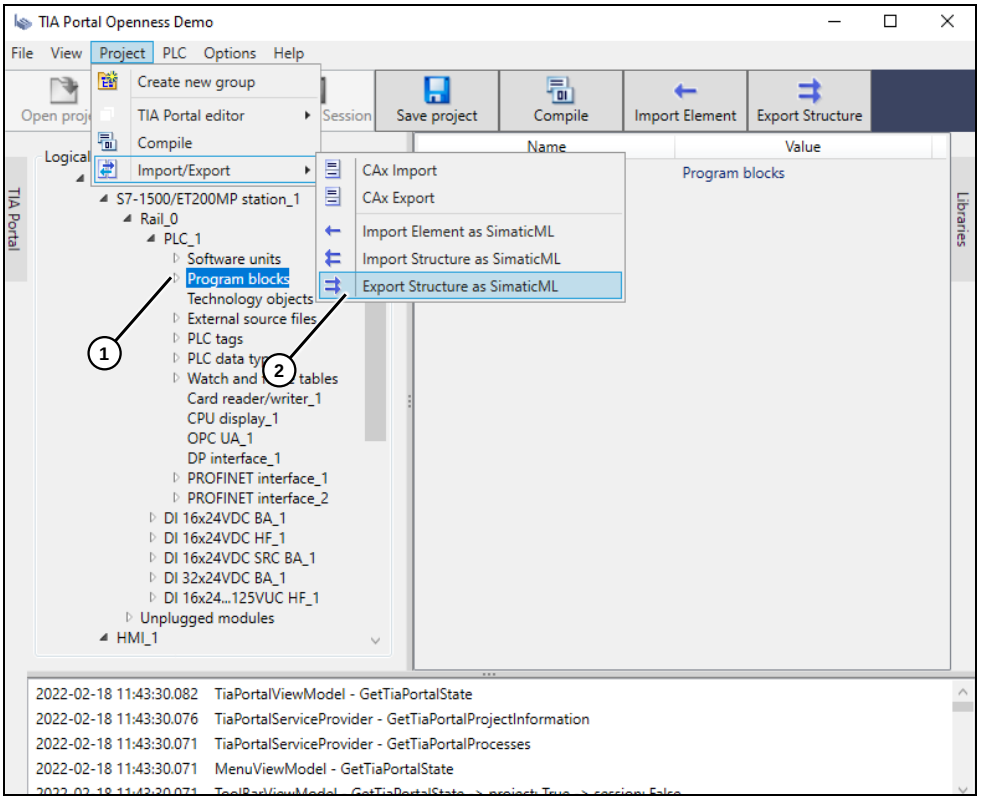


Table 6-19

No.	Description
1.	Highlighted area that will be exported as a structure.
2.	Starts the export via the menu "Project > Import/Export > Export Structure as SimaticML" or via the toolbar.

Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class

```
BaseProjectViewModel: ProjectServiceProvider =
_moduleProvider.GetService(typeof(IProjectServiceProvider)) as
IProjectServiceProvider;
```

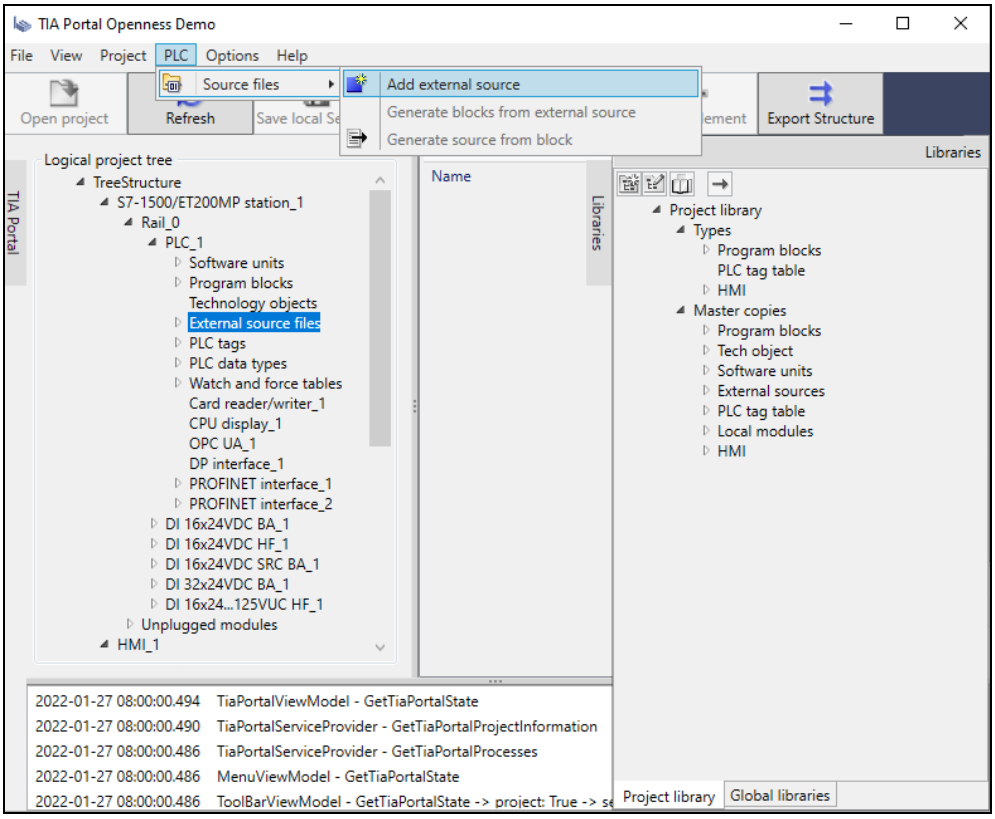
Depending on the view selected, the method `ExportStructureAsync` is called in either the class LogicalProjectTreeViewModel or PhysicalProjectTreeViewModel. In that method, the method of the same name is called on the service instance.

```
ProjectServiceProvider.ExportStructureAsync((string)SelectedItem.
Header, (Guid)SelectedItem.Tag, LogicalTreeView);
```

6.11 "PLC" menu

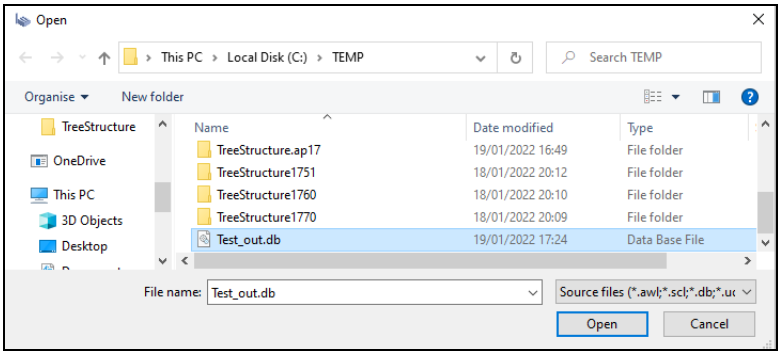
6.11.1 Add external source

Figure 6-44



If you highlighted "External source files" in the project tree, you can add a file via the menu "PLC -> Source files -> Add external source". A file selection dialog will first open in which you can select a file of type *.awl, *.scl, *.db or *.udt. When you open the selected file, it will be added as a new file.

Figure 6-45



Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

In the class `BaseProjectViewModel`, the method `AddExternalSource` is called in which, on the service instance, the call `ProjectServiceProvider.AddExternalSourceAsync(fileInfos, (string)SelectedItem.Header, (Guid)SelectedItem.Tag, LogicalTreeView);` is executed. The FileInfo of the selected file, the name of the highlighted element (see Figure 6-44) and the GUID of the selected element are passed as parameters. The fourth parameter in the call indicates that we are working on the logical project tree and that the project item should be found within it.

`GetProjectItem(header, tag, logical)` loads the project item from the corresponding tree view (logical or physical view). For the TiaPortal instance, an exclusive access operation with `tiaPortal?.ExclusiveAccess ("Import element")` is requested.

```
var projectItem = GetProjectItem(header, tag, logical);
if (projectItem != null)
{
  var destinationItem = projectItem.DeviceItem;
  if (destinationItem != null)
  {
    var tiaPortal = _tiaPortalServiceProvider.GetTiaPortal() as
                                                          TiaPortal;
    using (var plcService = new PlcService(_traceLogService))
    {
      foreach (var fileInfo in sourceFileInfos)
      {
        using (var exclusiveAccess = tiaPortal?.ExclusiveAccess
                                              ("Import element"))
        {
          using (var transaction = exclusiveAccess?.Transaction
                                    (CurrentProject, "Import element"))
          {
            plcService.AddExternalSource(destinationItem as
                                    PlcExternalSourceGroup, fileInfo);
            transaction?.CommitOnDispose();
            result = true;
          }
        }
      }
    }
  }
}
```

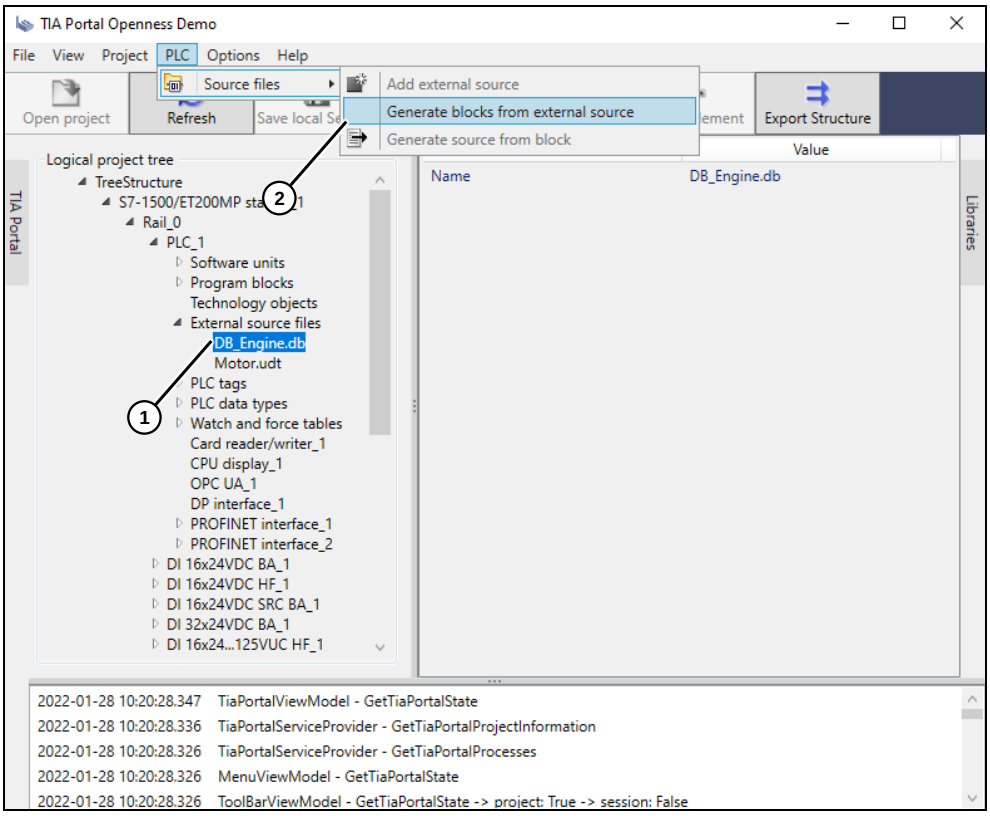
The method `AddExternalSource` is run on the instance of PlcService `using (var plcService = new PlcService(_traceLogService))`. The method checks whether the project was already assigned an external source file with this name and, if so, it is deleted. Then the selected file is generated as a new external source and then added.

```
var temp =
plcExternalSourceGroup.ExternalSources.Find(Path.GetFileName(externa
lSourceFileInfo.FullName));

temp?.Delete();
```

```
plcExternalSourceGroup.ExternalSources.CreateFromFile(Path.GetFileNa
me(externalSourceFileInfo.FullName),
externalSourceFileInfo.FullName);
```

6.11.2 Generate blocks from external source

Figure 6-46:



Following the sequence diagram (see Figure 6-7), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

In the class `BaseProjectViewModel`, the method `GenerateBlockFromSource` is called in which, on the service instance, the call

`ProjectServiceProvider.GenerateBlockFromSourceAsync((string)SelectedItem.Header, (Guid)SelectedItem.Tag, LogicalTreeView);` is executed.

The name and GUID of the highlighted element (see Figure 6-46:, item 1) in the project tree, together with the information as to whether the logical or physical project tree is used, are used to search for the project item.

```
var projectItem = GetProjectItem(header, tag, logical);
if (projectItem != null)
{
  var destinationItem = projectItem.DeviceItem;
  if (destinationItem != null)
  {
    using (var plcService = new PlcService(_traceLogService))
    {
      plcService.GenerateBlockFromSource(destinationItem as
                                                PlcExternalSource);
      result = true;
    }
```

```
    }
}
```

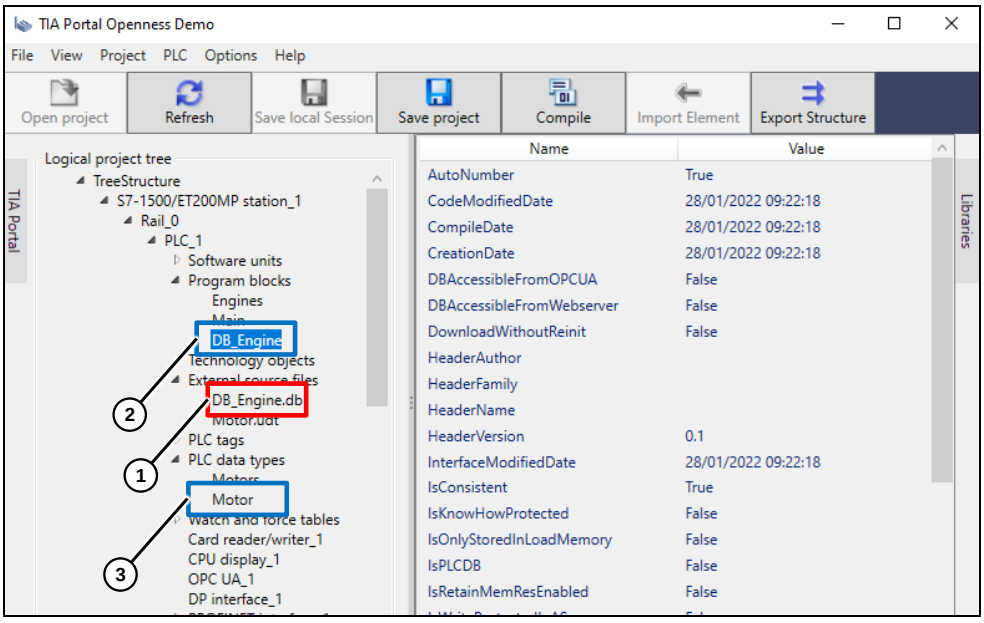
Table 6-20

No.	Description

1.	Highlighted project item of type "External source file" from which a block will be generated. The external source DB_Engine, for example, contains one type and one data block.
	`TYPE "Motor"`
	`VERSION : 0.1`
	`STRUCT`
	`Start : Bool;`
	`Stop : Bool;`
	`Temperature : Real;`
	`RPM : Real;`
	`State : Bool;`
	`END_STRUCT;`
	`END_TYPE`
	
	`DATA_BLOCK "DB_Engine"`
	`{ DB_Accessible_From_OPC_UA := 'FALSE' ;`
	`DB_Accessible_From_Webserver := 'FALSE' ;`
	`S7_Optimized_Access := 'TRUE' }`
	`VERSION : 0.1`
	`NON_RETAIN`
	`VAR`
	`Motor_1 : "Motor";`
	`Motor_2 : "Motor";`
	`Motor_3 : "Motor";`
	`Motor_4 : "Motor";`
	`END_VAR`
	`BEGIN`
	`END_DATA_BLOCK`

2.	Generate target blocks from the source file.

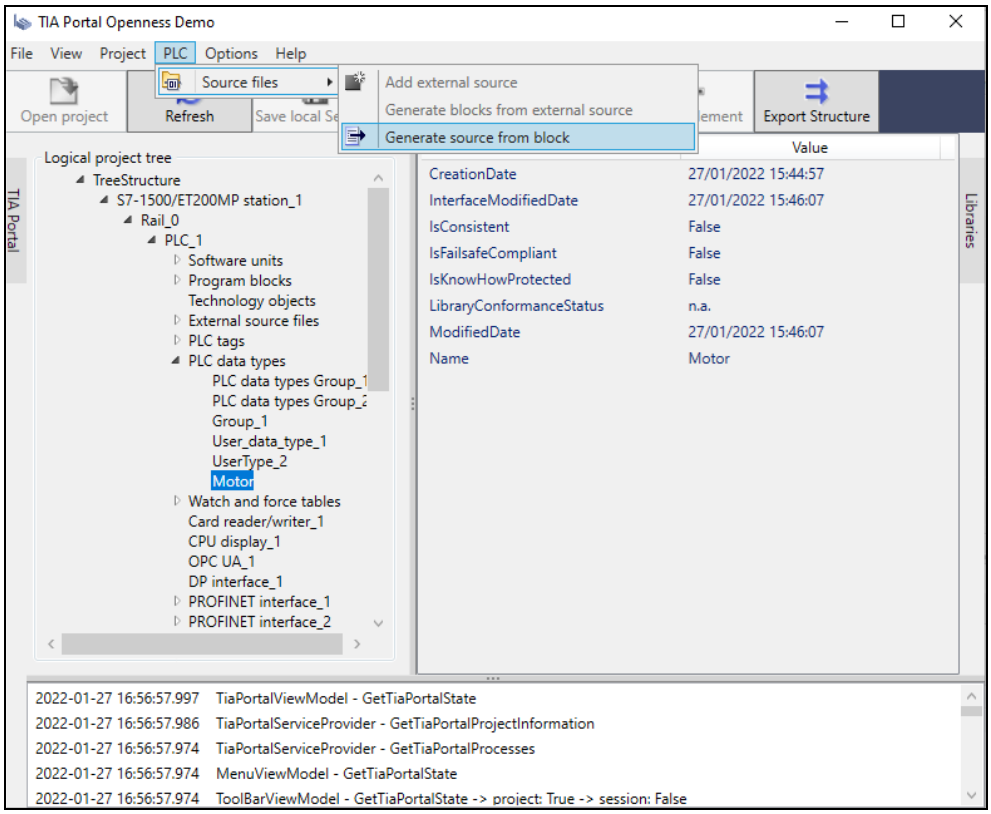
Figure 6-47



The target blocks are generated from the source file (see also Figure 6-47, item 1). The types from the external source (see Table 6-20, item 1) are assigned to the corresponding areas (see Figure 6-47, items 2 and 3).

6.11.3 Generate source from block

Figure 6-48:



For example, if you highlighted a PLC data type in the project tree, you can use the menu "PLC -> Source files -> Generate source from block" (see Figure 6-48:) to create a source file from this data type. The source file can be reused in another project.

First the Save File dialog opens (see Figure 6-49:). Here, depending on the highlighted element, you can select the file type *.awl, *.scl, *.db or *.udt and enter the file name.

Following the sequence diagram (see Figure 6-8), a service instance is loaded via the ModuleProvider in the method `InitProjectServiceProvider` of class `BaseProjectViewModel`: `ProjectServiceProvider = _moduleProvider.GetService(typeof(IProjectServiceProvider)) as IProjectServiceProvider;`

In the class `BaseProjectViewModel`, the method `GenerateSourceFromBlock` is called in which, on the service instance, the call

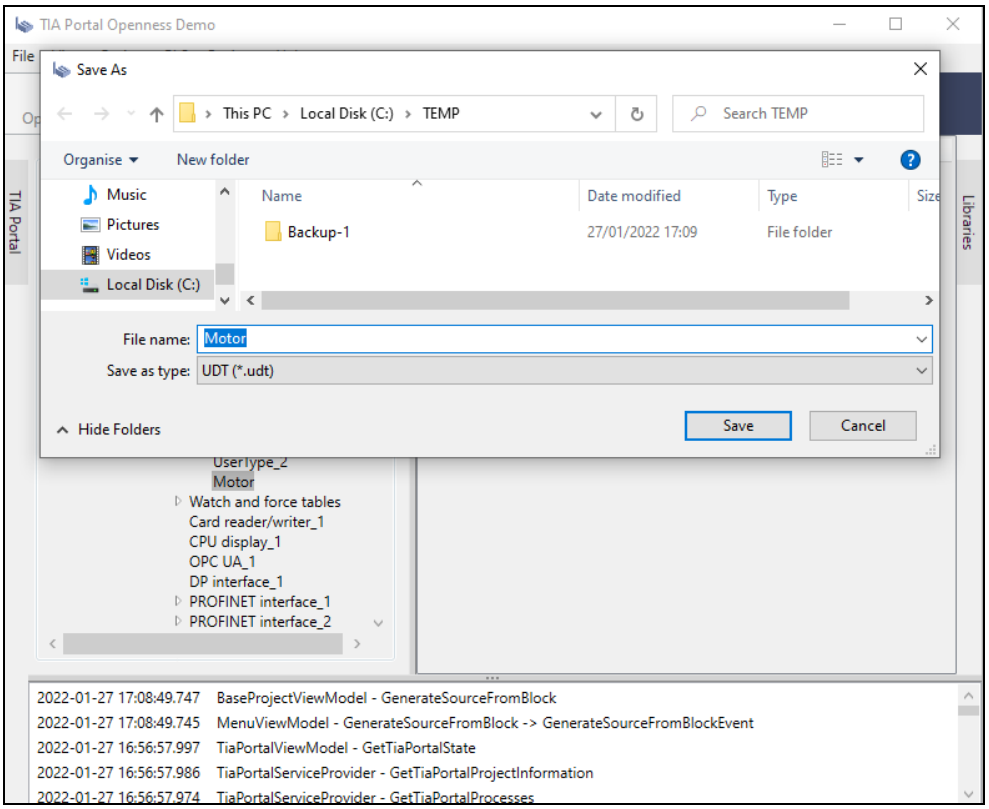
`ProjectServiceProvider.GenerateSourceFromBlockAsync(destinationFileInfo, (string)SelectedItem.Header, (Guid)SelectedItem.Tag, LogicalTreeView);` is executed. The parameters for searching for the project item are the file info of the target file, the name of the highlighted element, its GUID, and the information on whether the logical or physical project tree is to be used.

```
var projectItem = GetProjectItem(header, tag, logical);
if (projectItem != null)
{
  var blockItem = projectItem.DeviceItem as
                                              IEngineeringInstance;
  var blockAsSource = projectItem.DeviceItem as
                                              IEngineeringInstance;
  if (blockItem != null)
  {
    do
    {
       blockItem = blockItem.Parent;
    }
    while (!(blockItem is PlcSoftware));
    using (var plcService = new PlcService(_traceLogService))
    {
      plcService.GenerateSourceFromBlock(blockItem as
            PlcSoftware, blockAsSource, destinationFileInfo, true);
      result = true;
    }
  }
}
```

The actual API call is executed in the service method `plcService.GenerateSourceFromBlock`.

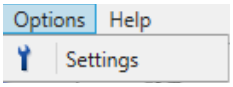
```
if (blockAsSource is PlcBlock plcBlock)
{
  if (withDependencies)
  {
    plcSoftware.ExternalSourceGroup.GenerateSource(new[]
                                  { plcBlock }, destinationFileInfo,
                                  GenerateOptions.WithDependencies);
  }
  else
  {
    plcSoftware.ExternalSourceGroup.GenerateSource(new[]
                                  { plcBlock }, destinationFileInfo,
                                  GenerateOptions.None);
  }
}
if (blockAsSource is PlcType plcType)
{
  plcSoftware.ExternalSourceGroup.GenerateSource(new[]
                                  { plcType }, destinationFileInfo);
}
```

Figure 6-49:



6.12 "Options" menu

Figure 6-50



The application's settings let you define the values for a number of parameters that the application automatically loads and uses when running the corresponding functions. The settings can be changed at any time. It is only necessary to restart the application following a change if you wish to use a different version of TIA Portal and/or the Openness API. All other settings are only loaded when used.

6.12.1 Settings

Figure 6-51

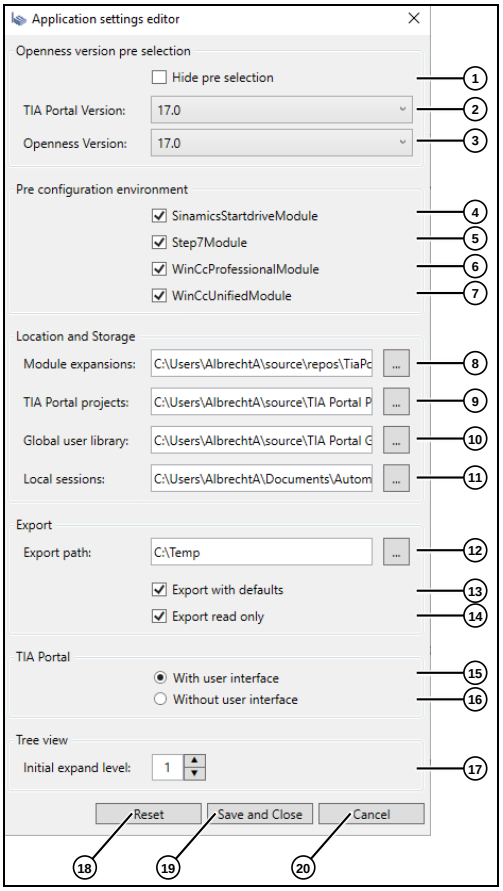
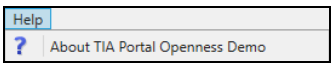


Table 6-21

No.	Description
1.	This setting defines whether or not to display the dialog for selecting your TIA Portal and Openness versions when the application launches (see Assembly Resolve).
2.	Installed TIA Portal version selector. The selected version will be loaded when the application starts (see Assembly Resolve).
3.	Installed Openness API version selector. The selected version will be loaded when the application starts (see Assembly Resolve).
4.	Decides whether or not to load the SinamicsStartdriveModule. You should only enable loading of this module if the Sinamics Startdrive software is installed on your system.
5.	Decides whether or not to load the Step7Module. You should only enable loading of this module if the STEP 7 software is installed on your system.
6.	Decides whether or not to load the WinCcProfessionalModule. You should only enable loading of this module if the WinCC Professional software is installed on your system.
7.	Decides whether or not to load the WinCcUnifiedModule. You should only enable loading of this module if the WinCC Unified software is installed on your system.
8.	This path specifies which directory the module expansions should be loaded from.
9.	Sets the default project directory. New projects are saved in this directory. When a project is opened, this path is used for the file selection dialog.
10.	Sets the default directory for global user-defined libraries. New global user-defined libraries are saved in this directory. When a global user-defined library is opened, this path is used for the file selection dialog.
11.	Sets the default directory for local sessions. This path will be used for the file selection dialog when opening a local session.
12.	Sets the default export directory.
13.	Includes default values when exporting.
14.	Includes write-protected values when exporting.
15.	Causes a new TIA Portal instance to launch with user interface.
16.	Causes a new TIA Portal instance to launch without user interface.
17.	This value defines how many levels the project and library trees will expand to.
18.	Undoes all changes and shows the values that were populated when the dialog opened.
19.	Saves all changes and closes the dialog.
20.	Closes the dialog without saving.

6.13 "Help" menu

Figure 6-52



6.13.1 About TIA Portal Openness Demo

Figure 6-53

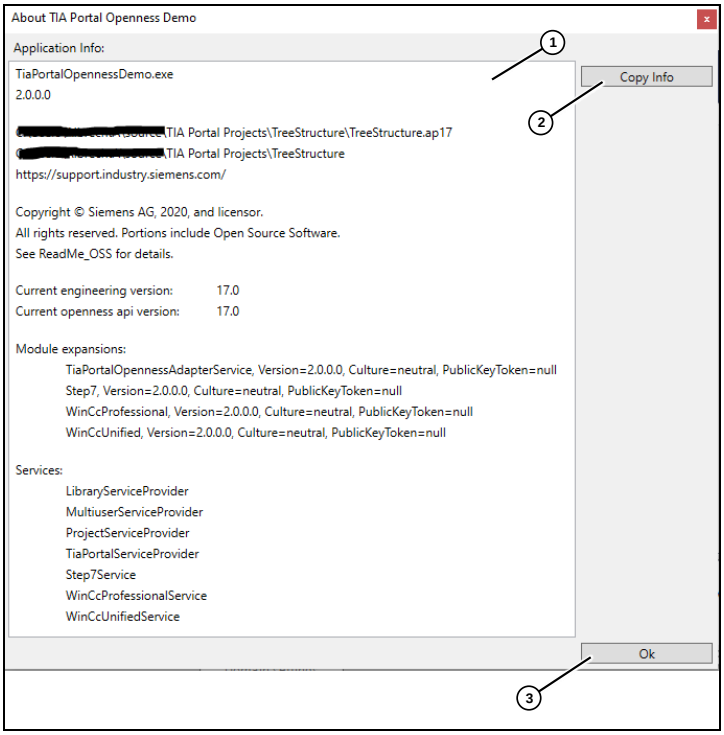


Table 6-22

No.	Description
1.	List of runtime information such as loaded engineering and Openness API version, loaded module expansions, or available services for the TIA Portal Openness Demo Application.
2.	Copies the list of runtime information together with the entire trace log to the clipboard.
3.	Close the About dialog.

7 Appendix

7.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:

siemens.com/SupportRequest

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:

siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for iOS and Android:

support.industry.siemens.com/cs/ww/en/sc/2067

7.2 Industry Mall

The Siemens Industry Mall is the platform on which the entire siemens Industry product portfolio is accessible. From the selection of products to the order and the delivery tracking, the Industry Mall enables the complete purchasing processing – directly and independently of time and location:
mall.industry.siemens.com

7.3 Links and literature

Table 7-1

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the article page of the application example https://support.industry.siemens.com/cs/ww/en/view/108716692
\3\	Support request http://www.siemens.com/automation/support-request
\4\	Why is your TIA Portal Openness application not working as expected? http://support.industry.siemens.com/cs/ww/en/view/109251656
\5\	When using a TIA Portal Openness application, why do you get the error message "Cannot connect to TIA Portal"? http://support.industry.siemens.com/cs/ww/en/view/109038214
\6\	System manual https://support.industry.siemens.com/cs/ww/en/view/109477163

7.4 Change documentation

Table 7-2

Version	Date	Change
V1.0	02/2015	First version
V1.0	09/2015	Fixed minor errors
V1.1	12/2016	Version for TIA Portal V14
V1.2	05/2017	Version for TIA Portal V14 SP1
V1.3	02/2018	Version for TIA Portal V15
V1.4	05/2019	Version for TIA Portal V15.1
V1.5	03/2023	Version for TIA Portal V17