SIEMENS

SIMATIC

TIA Portal Openness: API for automation of engineering workflows

System Manual

Online documentation

11/2023
Online documentation

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens Aktiengesellschaft. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens Aktiengesellschaft
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

Document order number: Online documentation
Ⓟ 12/2023 Subject to change

Table of contents

Cybersecurity information

1

Cybersecurity information

Siemens provides products and solutions with industrial cybersecurity 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 cybersecurity concept. Siemens' products and solutions only form 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 cybersecurity measures that may be implemented, please visit

https://www.siemens.com/global/en/products/automation/topic-areas/industrial-cybersecurity.html (https://www.siemens.com/global/en/products/automation/topic-areas/industrial-cybersecurity.html)

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 Cybersecurity RSS Feed under

https://new.siemens.com/global/en/products/services/cert.html (https://new.siemens.com/global/en/products/services/cert.html)

Readme TIA Portal Openness

2

2.1 Deprecated features in TIA Portal Openness

Introduction

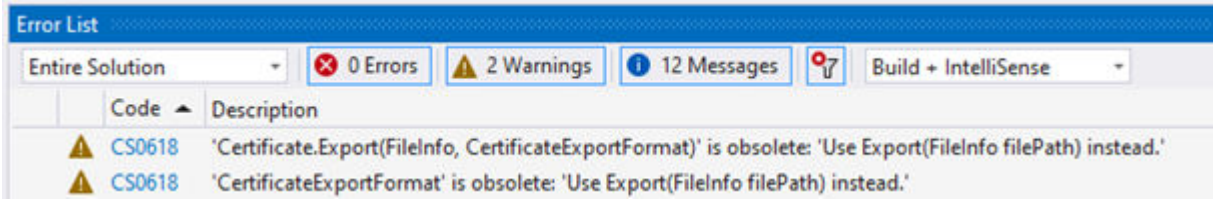
In general, TIA Portal Openness follows a long-term stable API strategy. There are isolated situations where APIs must be reworked:

Terminology:

Product release	TIA Portal Openness version	
TIA Portal V18	API V15.1	old API
	API V16	old API
	API V17	old API
	APIV18	new API
TIA Portal V19	API V16	old API
	API V17	old API
	API V18	old API
	APIV19	new API

Static object model parts

Deprecated features are announced for an API version and will be removed in the next API version of the product. The old APIs in the next product release are not affected.
For static object model parts, deprecated features are marked as obsolete. Using those features in code generates a compiler warning, but they still work in that version.
Example of a compiler warning:



Example of highlighted obsolete features in code:

```
private static void ExportCertificate(Siemens.Engineering.Security.Certificate certificate)
{
FileInfo fileInfo = new FileInfo($"C:\\Temp\\{certificate.SubjectCommonName}.cer");
certificate.Export(fileInfo, Siemens.Engineering.Security.CertificateExportFormat.Cer);
}
```

Consider adapting deprecated code as follows.

Deprecated in API V18

- Class: Siemens.Engineering.AddIn.VersionControl.InitialPreExportInfo
  Deprecated and substituted by Siemens.Engineering.AddIn.VersionControl.PreExportInfo
- Class: Siemens.Engineering.AddIn.VersionControl.InitialPostExportInfo
  Deprecated and substituted by Siemens.Engineering.AddIn.VersionControl.PostExportInfo
- Class: Siemens.Engineering.AddIn.VersionControl.SyncPreExportInfo
  Deprecated and substituted by Siemens.Engineering.AddIn.VersionControl.PreExportInfo
- Class: Siemens.Engineering.AddIn.VersionControl.SyncPostExportInfo
  Deprecated and substituted by Siemens.Engineering.AddIn.VersionControl.PostExportInfo
- Class: Siemens.Engineering.AddIn.VersionControl.VciInitialExportAddInContext
  Deprecated and substituted by Siemens.Engineering.AddIn.VersionControl.ExportContext
- Class: Siemens.Engineering.AddIn.VersionControl.VciSyncExportAddInContext
  Deprecated and substituted by Siemens.Engineering.AddIn.VersionControl.ExportContext
- Class: Siemens.Engineering.AddIn.VersionControl.VciRepositoryAddIn
  Deprecated and substituted by
  Siemens.Engineering.AddIn.VersionControl.VciWorkspaceRepositoryAddIn
- Class: Siemens.Engineering.AddIn.VersionControl.VciRepositoryAddInProvider
  Deprecated and substituted by
  Siemens.Engineering.AddIn.VersionControl.VciWorkspaceRepositoryAddInProvider
- Class: Siemens.Engineering.AddIn.VersionControl.VciInitialExportSupport
  Deprecated and substituted by
  Siemens.Engineering.AddIn.VersionControl.ExportWorkflowSupport
- Class: Siemens.Engineering.AddIn.VersionControl.VciSyncExportSupport
  Deprecated and substituted by
  Siemens.Engineering.AddIn.VersionControl.ExportWorkflowSupport
- Class: Siemens.Engineering.AddIn.VersionControl.VciWorkflowAddInSupport
  Deprecated and substituted by
  Siemens.Engineering.AddIn.VersionControl.VciWorkspaceRepositoryWorkflowAddIn
- Class: Siemens.Engineering.HmiUnified.RuntimeSettings.HmiRuntimeSetting

Deprecated property	Relocated
OperateAsOpcServer { get; set; }	Moved to class Siemens.Engineering.HmiUnified.RuntimeSettings.HmiOpcUaServerRuntimeSettings

- Class: Siemens.Engineering.HmiUnified.UI.Parts.HmiTrendPartBase

Deprecated property	Relocated
TrendMode { get; set; }	Moved from base class to derived classes Siemens.Engineering.HmiUnified.UI.Parts.HmiTrendPart and Siemens.Engineering.HmiUnified.UI.Parts.HmiFunctionTrendPart

- Class: Siemens.Engineering.SiVArc.ScreenRule

Deprecated property	Deprecated
ScreenLibraryItem { get; set; }	Deprecated without substitution.

Deprecated in API V19

- Class: Siemens.Engineering.HW.MulticastableTransferArea

Deprecated property	Substituted by
Siemens.Engineering.HW.Address Address { get; }	Siemens.Engineering.HW.AddressComposition Addresses { get; }

- Class: Siemens.Engineering.Security.Certificate

Deprecated method	Substituted by
System.Void Export(System.IO.FileInfo filePath, Siemens.Engineering.Security.CertificateExportFormat exportFormat)	System.Void Export(System.IO.FileInfo filePath)

- Enum: Siemens.Engineering.Security.CertificateExportFormat
  Deprecated without substitution
- Enum: Siemens.Engineering.HW.WebserverUserPermissions2
  Deprecated without substitution
- Enum: Siemens.Engineering.HW.MaximumBufferedReceivedFrames
  Deprecated without substitution
- Enum: Siemens.Engineering.HW.PortType
  Deprecated and substituted by Siemens.Engineering.HW.PortConfiguration
- Exception class: Siemens.Engineering.SimaticMLVersionNotSupportedException
  Deprecated without substitution

Dynamic API parts

Deprecated features are announced for a TIA Portal version and will be removed in the next TIA Portal version. The old APIs and new API in the next product release are affected.
For dynamic API parts, a compiler warning cannot be generated for technical reasons. Consider the system manual for deprecated features in that area.
Example of a dynamic API part:

```
private static ulong GetPortValue(Siemens.Engineering.HW.DeviceItem item)
{
const string attributeName = "PortType";
ulong attributeValue = (ulong)item.GetAttribute(attributeName);
return attributeValue;
}
```

Consider adapting deprecated code as follows

Deprecated in TIA Portal V18 (all API versions)

Hardware parameter in 6ES7148-6JG00-0BB0/V5.1

Deprecated attribute name	Substituted by
PortType	PortConfiguration

2.2 Readme

Security measures for TIA Portal Openness applications

It is recommended

- to install a TIA Portal Openness application with admin rights to the programs folder.
- to avoid the dynamical loading of program parts like assemblies or dlls from the users area.
- to run the TIA Portal Openness application with user rights.

Hardware parameters

A description of hardware parameters is available in the installation folder of TIA Portal at Siemens\Automation\Portal V*\PublicAPI\V*\HW Parameter Description\Openness_hardware_parameter_description.pdf

Attributes can have dependencies and influence each other. Therefore, if an attribute of a device item or channel has been changed, all attributes of the device item or channel must be verified.

Note

V* refers to adapted path according to the installed version of TIA Portal.

Copying a TIA Portal Openness application

When you copy an executable TIA Portal Openness application, it may occur under certain circumstances that the directory path in which the TIA Portal Openness application was originally created is read out by the TIA Portal Openness application.

Remedy:

If you have copied the TIA Portal Openness application to a new directory, open and close the properties dialog to update the Windows cache.

Support of specific features in a TIA Portal project

Failsafe

When you are using TIA Portal Openness there are restrictions regarding failsafe. Please consider the documentation "SIMATIC Safety - Configuring and Programming" for further information.

Test Suite

Openness Support for TIA Portal Test Suite is documented in the respective online help from Test Suite V19

Improvement of the TIA Portal Openness performance

To achieve the maximum performance of TIA Portal Openness you can switch off the global search feature of the TIA Portal. To switch off the global search use the GUI or the TIA Portal Openness API call. When the TIA Portal Openness application is finished the global search could be switched on again. Although this is improving the performance, all TIA Portal Openness features work fine even with global search switched on.

Thread-safe program code

Take care that your code is thread-safe, an event appears in a different thread.

Export behaviour of screen items with style enabled

Export of a screen items with style enabled will not export the attributes of the style item, but those of the screen item before activating the style. If a style is selected and UseDesignColorSchema for the screen item is checked, the screen item fetches the attribute values from the style in the user interface but the attribute values of the screen item that were set before selecting the style are still stored in the database for this screen item. TIA Portal Openness exports these actual values that are stored in the database.

After disabling and enabling the style and exporting the screen item again, the same attribute values will be exported for the screen item like in the style item. If UseDesignColorSchema is unchecked, the attribute values of the selected style item are saved to the database for that screen item.

This problem can be solved by following the steps below:

1. Associate the screen item to the style item:
   - The database contains the attribute values before activating the style.
   - The user interfaces fetches attributes from the style item directly.
2. Export the screen item associated to the style item:
   - The XML file contains the attribute values from the database which are those before activating style.
3. Disable the UseDesignColorSchema:
   - The attribute values of style item are written in the attributes of the screen item in the database.

4. Enable the UseDesignColorSchema:
   – The attribute values of the screen item in the database are not changed and are still the ones from 3.
   – The user interfaces fetches attributes from the style item directly.
5. Export the screen item associated to the style item:
   – The XML file contains the attribute values from the database which were set at step 3, which are the same as the values in the style item.

Importing ASi slaves via AML

If one of the following ASi slaves is imported via an aml-file the firmware version of the device item will be set to V13.0 in all cases:

- ASIsafe FS400 RCV-B: 3SF7 844-*B***-***1
- ASIsafe FS400 RCV-M: 3SF7 844-*M***-***1
- ASIsafe FS400 TRX-M: 3SF7 844-*M***-**T0
- ASIsafe FS400 RCV-C: 3SF7 844-*T***-***1

Exporting and importing function keys

Function keys are synchronized during the import. If a function key is created in the global screen and the key is empty in the screen, the corresponding function key will use the global definition in all screens.

If you want to disable the global use of function keys after the import, define empty keys in the screens and import the screen types in the following order: Global screen, templates, screens.

If you want to ensure when exporting the screens that the global definition of a function key is not used by the template or by the global screen, create an empty function key in the screen. Select the required function key in the screen, then enable the "Use global assignment" property and disable it again.

Accessing a device while Online

Writing attributes of a device that is Online is not supported. Reading attributes is supported.

Disconnecting a subnet is not supported when the device is online.

Instance-specific attributes when importing blocks via TIA Portal Openness

In certain situations, the import rules can mean the loss of instance-specific attributes, such as start values, for example.

Writing access for OB block priority attribute

The Priority attribute name for write access for OB block is changed to PriorityNumber.

Information on specific features

Please See FAQ entries in Siemens Industry Online Support for further information concerning the following Openness functionalities:

- Archive/retrieve project
- Export/import watch tables

See also

Test Suite V18 (https://support.industry.siemens.com/cs/ww/en/view/109813414)

2.3 Major changes for long-term stability in TIA Portal Openness V19

Changes

If you have considered the hints concerning programming across versions and do not rebuild your Openness application to V19, your application will run without any restrictions on any computer even if only a TIA Portal V19 is installed.

If you rebuild your Openness application to V19, it is necessary to recompile your application using the Siemens.Engineering.dll of V19. In some cases, it might be necessary to adapt the code of your application.

CAx/AML Data exchange

Changes in behaviour of command line facility

With TIA Portal V19 onwards, the command line facility for CAx export and import is no longer available to the user.

Change in behaviour of Profinet / Ethernet port handling

With TIA Portal V19 onwards, Label value for Profinet / Ethernet interface ports would be exported without space irrespective its value in TIA Portal. For example: If Label = P1 R, AML file would have Label = P1R

Change in behavior of IO Link port handling

With the TIA Portal V18 Update 2 onwards, IO Link port shall be exchanged with Label value as 'C/Q<n>' where n is the port number. for example: C/Q1, C/Q2 etc.,
For AML exchange of IO Link configuration in TIA Portal V18 Update 2 (via S7-PCT), use "S7-PCT 3.5 SP3 Update 3" version or higher.

WebServerUserManagement

Changes in behaviour of removal/disable of existing Openness services

With the TIA Portal V19, the Centralized UMAC functionality will be active, but the WebServerUserManagement will be disabled for all PLC firmware versions from V3.1 onwards.

However, until TIA Portal V18, the WebServerUserManagement will be operational, while the Centralized UMAC functionality will be inactive for PLC firmware versions up to V3.0.

UMAC on PLC

Changes in behaviour of online legitimation call via UMAC on PLC

If the Openness application is running with a TIA Portal Openness API <= 18, the online legitimation via Openness is still possible via the download and upload configurations. But only if the used PLC is protected by a legacy ProtectionLevel legitimation.

With TIA Portal Openness API >= V19, all legitimation calls, especially the new UMAC on PLC, will be delivered to the event handler of the online legitimation event of the ConnectionConfiguration class. In case the OnlineAuthenticationConfiguration type is not handled there by the user code, there will be a second call to the callback method of the particular feature (e.g., download or upload). But there only the old protection mechanisms (until V18) can be handled.

Enforcement of strict password policies

Changes in behaviour of strict password policies for know how protection of blocks

With TIA Portal V19, the Strict password policies are applied while setting a password. This also applies to TIA Portal Openness calls, especially setting know-how protection. An exception will be thrown if the password does not follow the policy.

WinCC Unified Screen Editor

Changes in behaviour of MultilingualText items in context to WinCC Unified Screen Editor

All MultilingualText items are changed from unformatted to formatted in scope of the WinCC Unified Screen Editor since TIA Portal V18.
This means all texts must be set formatted from now on. Plain text will be rejected by throwing an exception. for example:

```
Language language = project.LanguageSettings.Languages.Find(new CultureInfo("en-US"));
MultilingualText multilingualToolTipText1 = ((HmiButton)screenItem1).ToolTipText;
MultilingualTextItem multilingualTextItem1 = multilingualToolTipText1.Items.Find(language);
multilingualTextItem1.Text = "<body><p>Modified button text from Openness</p></body>";
```

Support of Blocks/UDTs

Changes in schema to support NamedValueTypes usage in Blocks

In TIA Portal V19, the support for Named Value Types has been introduced (only for Software Units in S7-1500 PLCs).
In Openness, a new scope "NamedValueConstant" is introduced in the SimaticML schema from V19 to support the usage of a Named Value Type in Program Blocks or PLC Data Types. The Named Value Types which are used in PLC programming artifacts (Program Blocks, PLC

Data Types) are visible in the new scope "NamedValueConstant" during import/export via SimaticML.

```
<Access Scope="NamedValueConstant" UId="27">
<Constant Name="_.siemens.simatic.Named_value_type_1#UNDEFs" UId="28"/>
</Access>
<Token Text=";" UId="31"/>
<NewLine Num="2" UId="32"/>
```

2.4 Hints for writing long-term stable code

Version change

If you consider some hints for writing long-term stable code you will be able to use your application with other versions of the TIA Portal without modifying the code of your application.

Note

V* and *.ap* in document refer to the adapted path and extension according to the installed version of the TIA Portal.

Registry path and appconfig file

Modifications are necessary to change registry path and application configuration file, for instance:
“C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\V14 SP1\Siemens.Engineering.dll” has to be changed to
“C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Siemens.Engineering.dll”

To write long-term stable code, the registry path should be configurable and the application configuration file must be updated.

Installation path

Modifications are necessary to change the installation path of TIA Portal, for instance:
“C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\V14 SP1\Siemens.Engineering.dll” has to be changed to
“C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Siemens.Engineering.dll”

To write long-term stable code, the installation path should be configurable.

Extensions of TIA Portal project files and libraries

Modifications are necessary to change the extensions of TIA Portal project file and of libraries, for instance:
*.ap14
has to be changed to
.ap

To write long-term stable code, the extensions of TIA Portal project files and libraries should be configurable.

Opening a project

To write long-term stable code, the `Projects.OpenWithUpgrade` method should be used instead of the `Projects.Open` method.

Hierarchy of results

The hierarchy and/or the order of result classes e.g. compile result, might change across versions.

To write long-term stable code, you should avoid making assumptions about the depth and order of specific results.

The class layout is actually considered long term stable, mention explicit type names CompilerResult, CompareResult, DownloadResult, UploadResult. There is also a new results class: TransferResult. Content, hierarchy and order follow what is presented on the TIA Portal user interface of the currently executed or installed TIA Portal.

What's new in TIA Portal Openness? 3

Compatibility and long term stability

- Siemens.Engineering.dll assemblies
  Because the Siemens.Engineering.dll assemblies of V16, V17 and V18 are included in the scope of delivery, applications based on V16, V17 and V18 also run in V19 without modification. To make use of the functions of V19, you must integrate the dll of V19 and recompile the application.
  The Siemens.Engineering.dll assemblies can be found in the installation directory under "PublicAPI\[version]\". For example, the V19 dll can be found as "C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V19\Siemens.Engineering.dll".

  Note

  V15.1 Siemens.Engineering.dll will not be available as part of TIA Portal Openness V19.

- Exporting Simatic ML files
  The Siemens.Engineering.dll assemblies V16, V17 and V18 will create Simatic ML files of TIA Portal version V19.
- Importing Simatic ML files
  Each Siemens.Engineering.dll assemblies is able to import Simatic ML files from that version and any supported version from a previous release. For example, Simatic ML files of V18 can be imported with Siemens.Engineering.dll assemblies V19.

Note

V* refer to the installed version of the TIA Portal Openness API.

For changes to the object model see TIA Portal Openness object model (Page 53) for further information.

New features

The following new features and innovations are available in TIA Portal Openness V19. For more details on the various topics, refer to the relevant sections of the product documentation.

- General
  - Import of any supported SimaticML version regardless which API version is used.
  - More flexible SimaticML import with options to handle project languages.
- Projects
  - Configure usage of S7-1500 blocks in virtual PLCs

- Hardware configuration
  - Create, read, update, and delete transfer areas for PLC-PLC direct data exchange.
  - Manage PLC system logging configuration.
  - Manage new PLC access control configuration for CPU firmware 3.1 (UMAC and access levels).
  - Change default language for OPC UA alarms and events in PLC multilingual support.
  - Bulk changing hardware parameters in the right ordering in all overloaded "SetAttributes" methods
- Hardware modules
  - New supported modules for parameter access: ET200pro Safety, ET200eco PN Safety, ET200MP Safety.
  - Extended parameter access for SCALANCE XC-200 (since V4.3), XP-200 (since V4.3), SC-600 (since V2.3).
- CAx data exchange
  - Import/Export CAx data via AutomationML and retrieve results via API instead of log file.
  - Support of additional attributes for hardware configuration exchange.
- Online scenarios
  - Get accessible online devices for download or upload.
  - Download PLC including Safety to SIMATIC memory card folder.
  - Handle UMAC user management data for download.
  - Provide UMAC credentials for online PLC access legitimation.
- PLC user programs
  - Read and write access for additional columns in data blocks.
  - Informative attribute on program blocks about support for virtual PLCs.
  - Extension of SimaticML schema to support usage of Named Value Types.
  - Named Value constants are supported during block export/import.
- Safety engineering
  - Configure serial numbers for unique identification of a F-PLC in Safety Administration Editor.
- UMAC
  - Configure alias name for project users.
- Technology objects
  - Support of groups for technology objects.
  - Import and export files for interpreter programs

- Startdrive
  - Connect a technology object to a Startdrive telegram.
  - Read hardware ID of a Startdrive telegram.
  - Support of new Startdrive devices.
  - Parameterization of 3rd party encoders.
- WinCC Unified
  - Access to additional object attributes and runtime settings.
- Test Suite Advanced
  - Configure OPC UA settings for system tests.
  - Configure mode for application tests.
  - Support for master copies.
- Version Control Interface (VC)
  - VCI setting to handle not active project languages during SimaticML import.
  - Initialize the status of the VCI object to the mapped file.
- Alarming and ProDiag
  - Manage system diagnostics in PLC configuration.
- Web server
  - Manage data access for web server for CPU firmware 3.1.

See also

Major changes in TIA Portal Openness V16 (Page 1639)
Major changes for long-term stability in TIA Portal Openness V17 (Page 1635)
Major changes for long-term stability in TIA Portal Openness V19 (Page 29)
Major changes in TIA Portal Openness V15.1 (Page 1641)
Major changes in TIA Portal Openness V15 (Page 1643)
Major changes in V14 SP1 (Page 1646)
Major changes in V14 (Page 1678)

Basics

4

4.1 Installation

4.1.1 Requirements for TIA Portal Openness

Requirements for using TIA Openness applications

- A product based on the TIA Portal is installed on the PC, for example, "STEP 7 Professional" or "WinCC Professional".
- The "TIA Portal Openness" is installed on the PC.
  See Installing TIA Portal Openness (Page 38)

Supported Windows operating systems

The following table shows which combinations of Windows operating system, TIA Portal and user application are mutually compatible:

Windows operating system	TIA Portal	User application
64-bit	64-bit	32-bit, 64-bit and "Any CPU"

Requirements for programming TIA Portal Openness applications

- Microsoft Visual Studio 2017 or later with .Net Framework SDK 4.8 and the Windows Classic Desktop package

Necessary user knowledge

- Knowledge as a system engineer
- Advanced knowledge of Microsoft Visual Studio 2017 or later with .Net Framework SDK 4.8
- Advanced knowledge of C# / VB.net and .Net Framework
- User knowledge of the TIA Portal

TIA Portal Openness remoting channels

The TIA Portal Openness remoting channels are registered as type IpcChannel with the "ensureSecurity" parameter set to "false".

Note

You should avoid registering another IpcChannel using a "ensureSecurity" parameter value other than "false" with a priority higher than or equal to "1".

The IpcChannel is defined with the following attributes:

Attribute	Settings
"name" and "portName"	Set to $"{Process.Name}_{Process.Id}" or $"{Process.Name}_{Process.Id}_{AppDomain.Id}" when registered in an AppDomain other than the application's default.
"priority"	Set with the default value of "1".
"typeFilterLevel"	Set to "Full".
"authorizedGroup"	Set to the NTAccount value string for the built-in user account (i.e. everyone).

See also

Adding users to the "Siemens TIA Openness" user group (Page 39)

4.1.2 Installing TIA Portal Openness

Introduction

The "TIA Portal Openness" is installed via TIA Portal setup program by selecting the TIA Portal Openness checkbox (under Options) during TIA Portal installation.

Requirements

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator rights.
- Running programs are closed.
- Autorun is disabled.
- WinCC and/or STEP 7 are installed.
- The version number of the "TIA Portal Openness" matches the version numbers of WinCC and STEP 7.

Note

If a previous version of TIA Portal Openness is already installed, the current version will be installed side by side.

Procedure

To install the TIA Portal Openness, ensure the TIA Portal Openness checkbox is selected during the installation of TIA Portal. Follow the below steps to check the TIA Openness installation.

1. Under Configuration menu, select the folder Options.
2. Check the TIA Portal Openness checkbox.
3. Click "Next" and select the required option.

Follow the installation procedure of TIA Portal to complete the TIA Portal Openness installation.

Result

"TIA Portal Openness" is installed on the PC. Moreover, the local user group "Siemens TIA Openness" is generated.

Note

You still do not have access to the TIA Portal with the "TIA Portal Openness" add-on package. You need to be a member of the "Siemens TIA Openness" user group (see Adding users to the "Siemens TIA Openness" user group (Page 39)).

4.1.3 Adding users to the "Siemens TIA Openness" user group

Introduction

When you install TIA Portal Openness on the PC, the "Siemens TIA Openness" user group is automatically created.

Whenever you access the TIA Portal with your TIA Portal Openness application, the TIA Portal verifies that you are a member of the "Siemens TIA Openness" user group, either directly or indirectly by way of another user group. If you are a member of the "Siemens TIA Openness" user group, the TIA Portal Openness application starts and establishes a connection to the TIA Portal.

Procedure

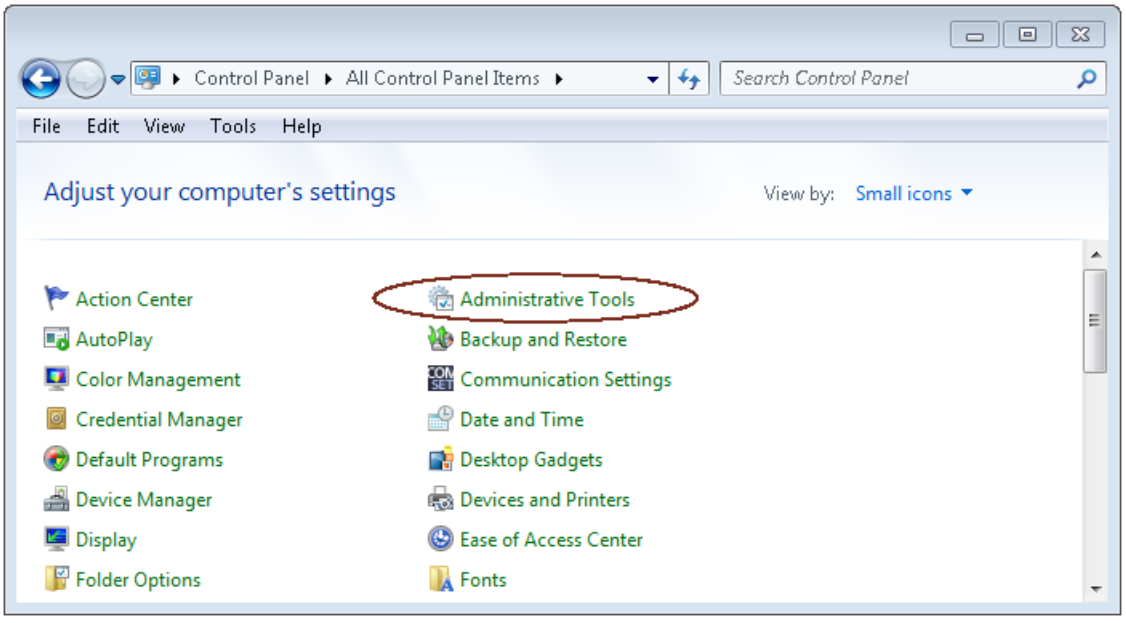
You add a user to the "Siemens TIA Openness" user group with applications from your operating system. The TIA Portal does not support this operation.

Note

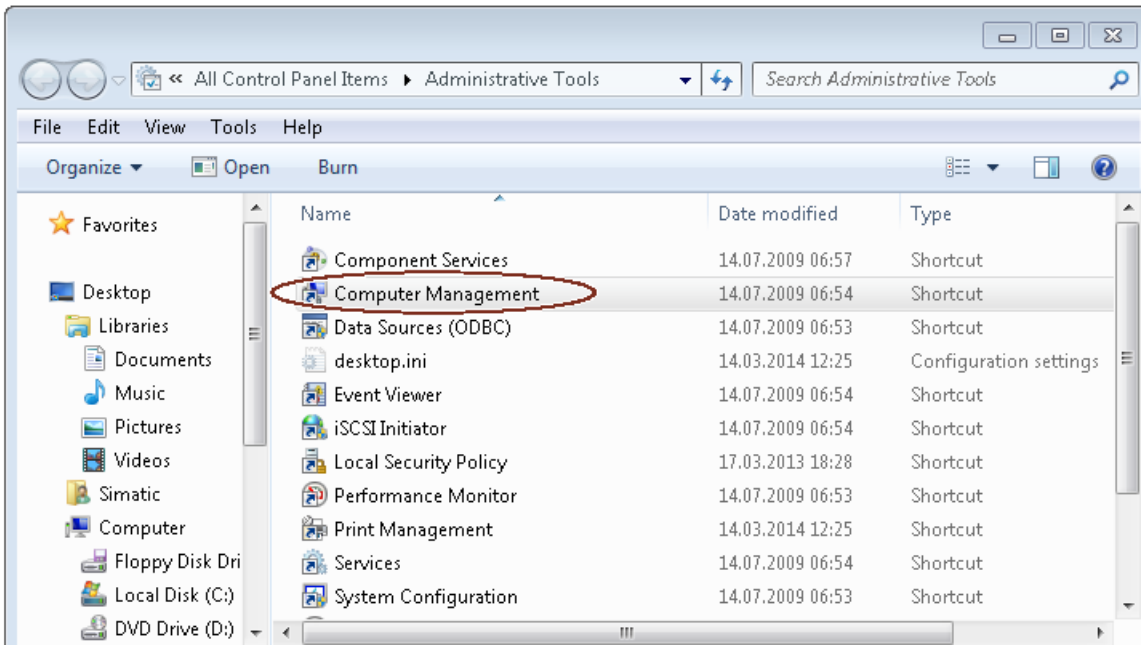
Depending on the configuration of your domain or computer, you may need to log on with administrator rights to expand the user group.

In a Windows 7 operating system (English language setting), for example, you can add a user to the user group as follows:

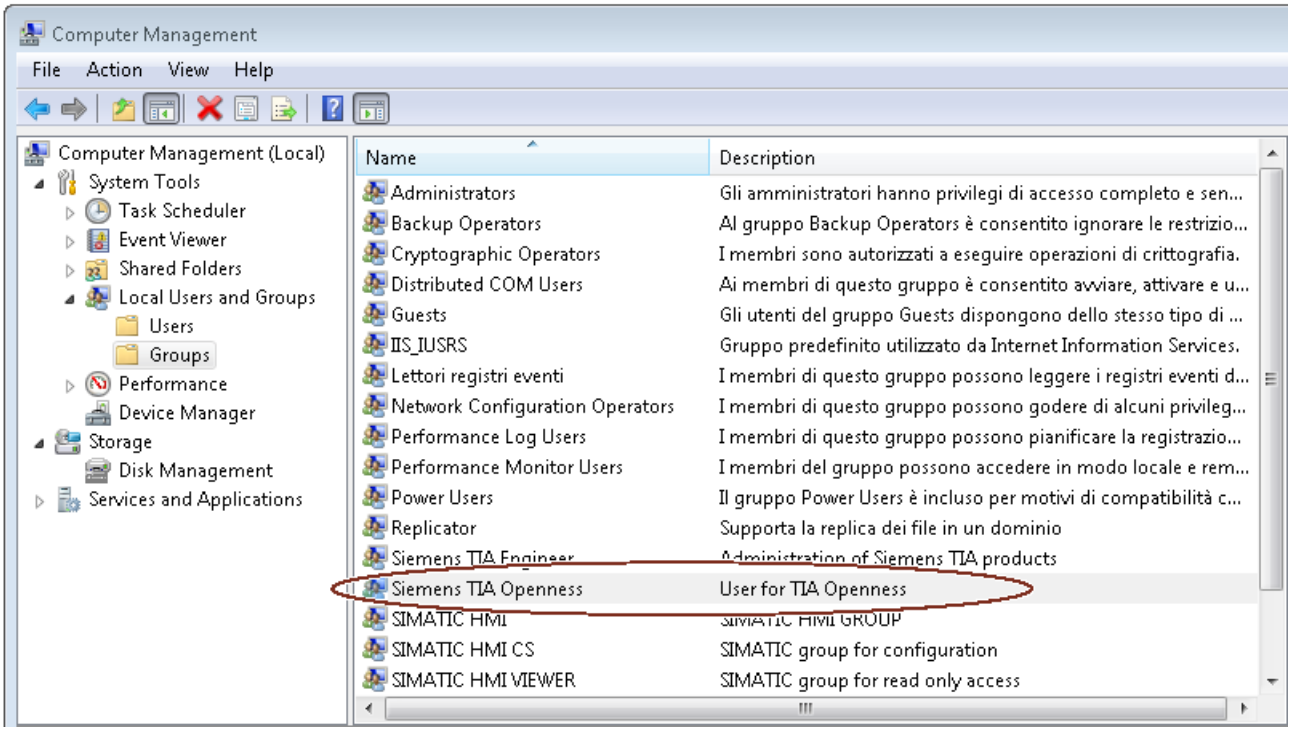
1. Select "Start" > "Control Panel".
2. Double-click "Administrative Tools" in the Control Panel.



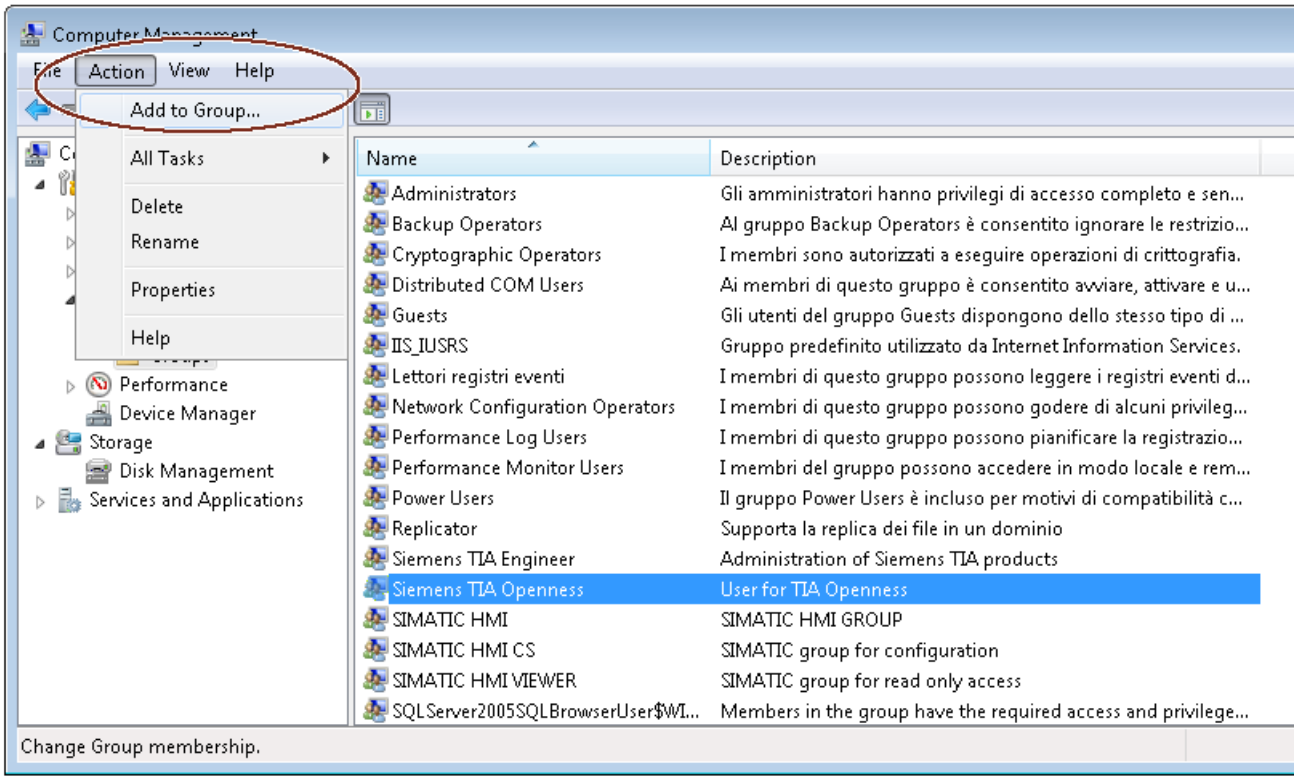
3. Click "Computer Management" to open the configuration dialog of the same name.



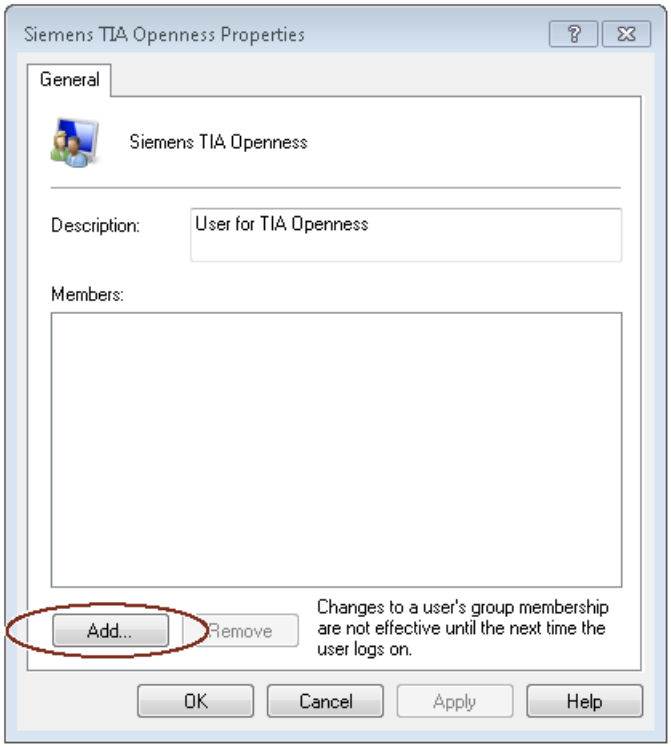
4. Select "Local Users and Groups > Groups", in order to display all created user groups.
5. Select the "Siemens TIA Openness" entry from the list of user groups in the right pane.



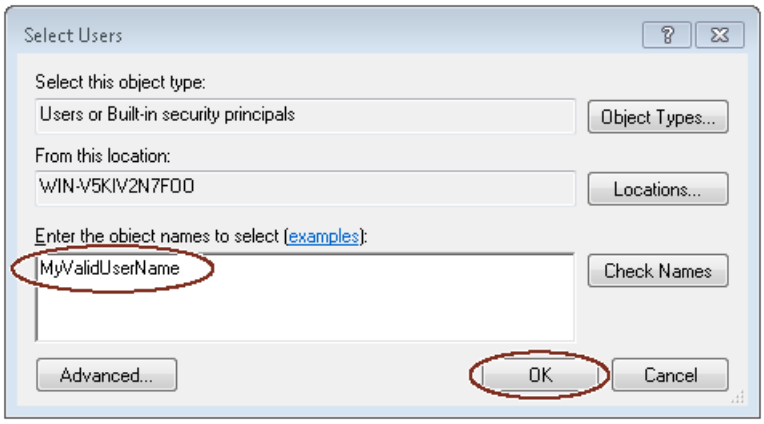
6. Select the "Action > Add to Group..." menu command.



The attributes dialog of the user group opens:



7. Click "Add".
   The selection dialog that opens displays the users that can be selected:



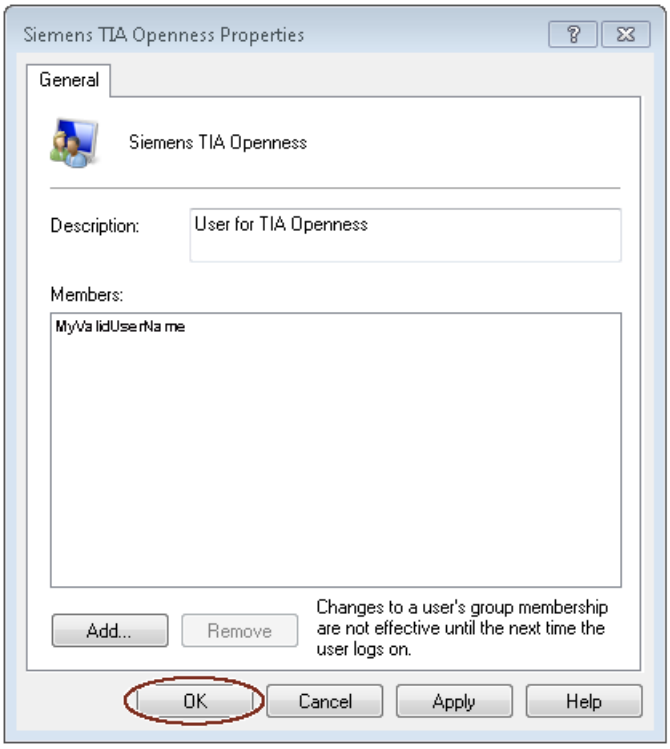
8. Enter a valid user name in the input field.

Note

Click "Check Names" to verify that the user entered has a valid user account for this domain or computer.

The "From this location" field displays the domain or computer name for the user name entered. For more information, contact your system administrator.

9. Confirm your selection with "OK".
   The new user is now displayed in the attributes dialog of the user group.



   You register additional users by clicking the "Add" button.

10. Click "OK" to end this operation.
11. Log on to the PC again for the changes to take effect.

4.1.4 Using Windows Service

Introduction

Windows Services are long-running executables that run without a user interface and are installed using an installation utility. It is possible to develop such a Windows service application that uses TIA Portal Openness.

To do so, it is necessary to create a project of type "Windows Service (.NET Framework)" in Microsoft Visual Studio.

Requirement

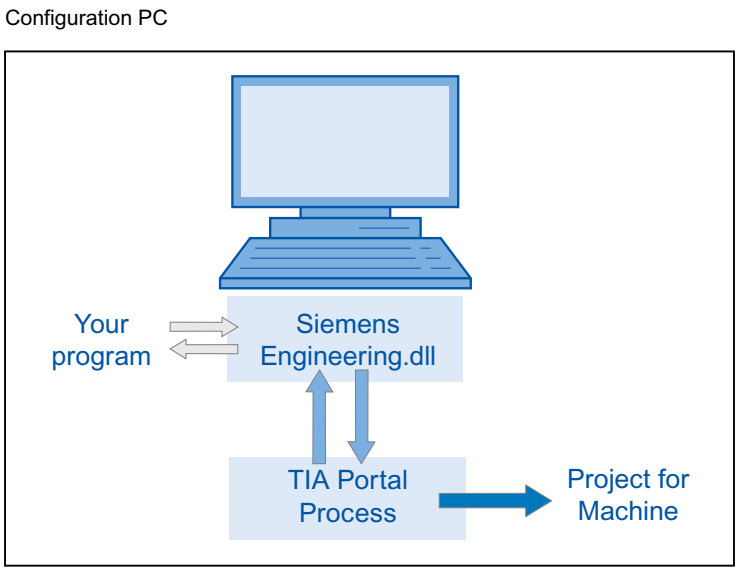
- The process executable of the Windows Service application needs to be whitelisted in the TIA Portal Openness Firewall beforehand. This can be done by creating the Windows registry entries on installation of the Windows Service, for example as part of a setup created with a Windows Installer project.
  For more information about whitelisting, see section "Addition of a whitelist entry without using the TIA Portal" in chapter TIA Portal Openness firewall (Page 88)
- It is not possible to access running TIA Portal processes or to start TIA Portal with graphical interface. Only a new TIA Portal instance without user interface can be created.
- It is highly recommended to run the Windows Service with credentials of a dedicated Windows user account. It is possible but not recommended to use the local service account or the network service account. The local system account is not supported due to security restrictions and will always throw a EngineeringSecurityException.
- The used Windows account (either service or user) needs to be part of the local Windows user group "Siemens TIA Openness".

See also

TIA Portal Openness firewall (Page 88)

4.1.5 Accessing the TIA Portal

Overview



Procedure

1. Set up the development environment to access and start the TIA Portal.
2. Instantiate the object of the portal application in your program to start the portal.
3. Find the desired project and open it.
4. Access the project data.
5. Close the project and exit the TIA Portal.

See also

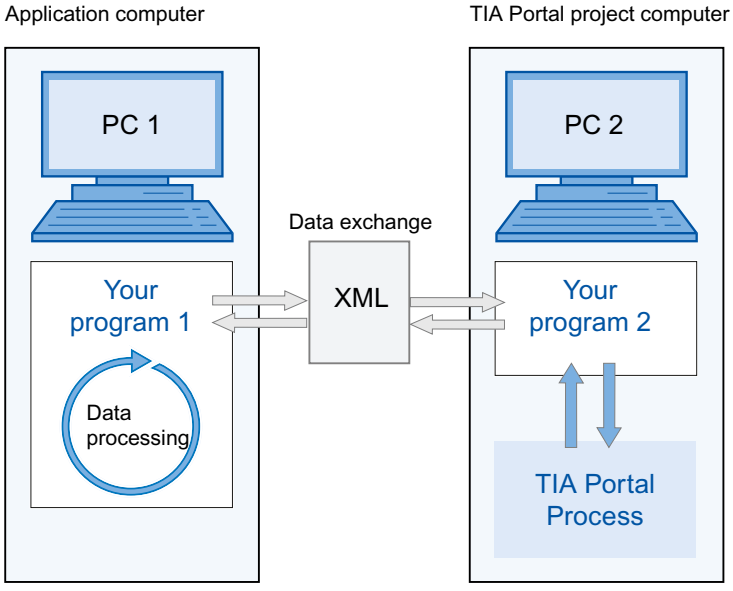
Connecting to the TIA Portal (Page 82)

Terminating the connection to the TIA Portal (Page 98)

4.1.6 Configurations

You can work with two variants of the TIA Portal Openness:

Application and the TIA Portal are on different computers



- Data exchange takes place by XML files. The XML files can be exported or imported by your programs.
- The data exported from the TIA Portal project to PC2 can be modified on PC1 and re-imported.

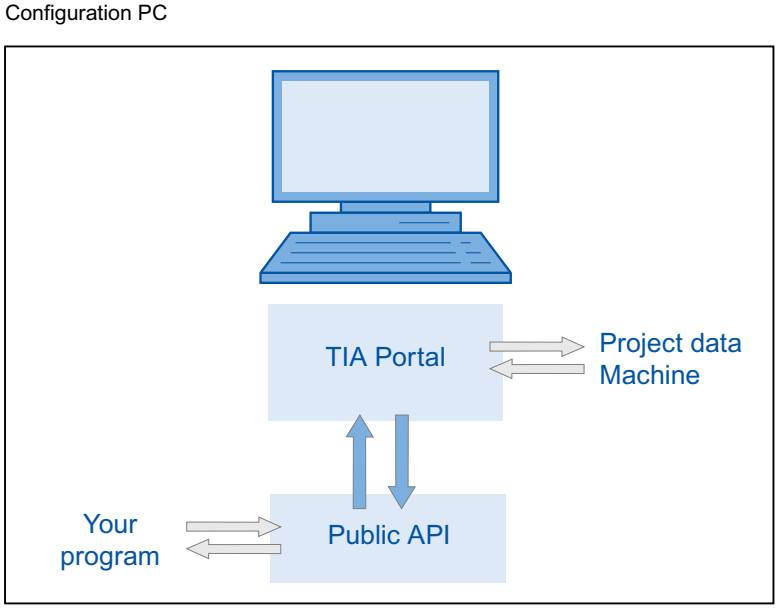
Note

You have to develop an executable program "Your Program 2" for PC2, such as "program2.exe". The TIA Portal runs with this program in the background.

Import and export of XML files takes place exclusively via the TIA Portal Openness API.

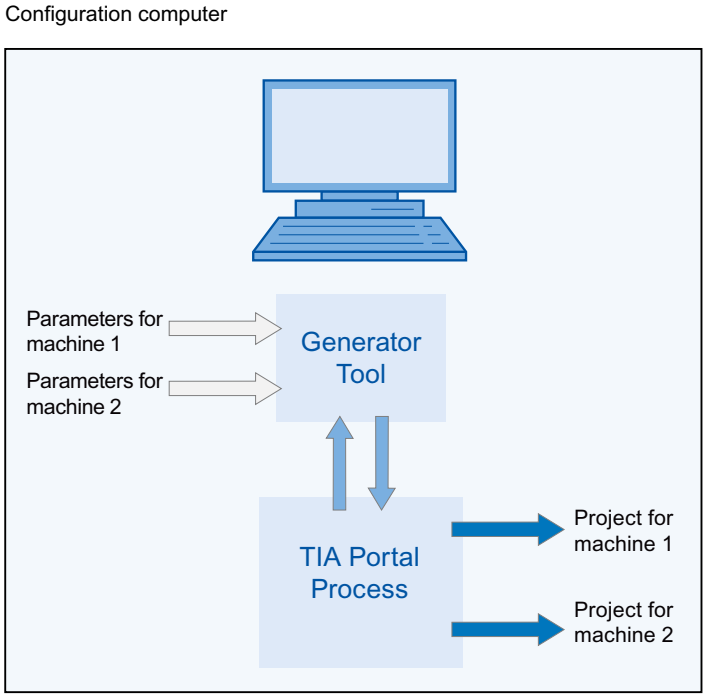
- You can archive exchanged files for verification purposes.
- Exchanged data can be processed at different locations and times.

Application and the TIA Portal are on the same computer



- Your program launches the TIA Portal either with or without the user interface. Your program opens, saves and/or closes a project. The program can also connect to a running TIA Portal.
- You can then use the TIA Portal functionality to request, generate and modify project data or to initiate import or export processes.
- The data is created under control of the TIA Portal processing and stored in the project data.

Typical application in modular mechanical engineering



- An efficient automation system is to be applied to similar machines.
- A project is available in the TIA Portal, which contains the components of all machine variants.
- The Generator tool controls the creation of the project for a specific machine variant.
- The Generator tool obtains the defaults by reading in the parameters for the requested machine variant.
- The Generator tool filters out the relevant elements from the overall TIA Portal project, modifies them if necessary and generates the requested machine project.

4.2 Openness tasks

4.2.1 Introduction

Introduction

TIA Portal Openness describes open interfaces for engineering with the TIA Portal. You will find further information about "TIA Portal Openness - Efficient generation of program code using code generators" in the SIEMENS YouTube channel (www.youtube.com/watch?v=Ki12pLbEcxs).

You automate the engineering with TIA Portal Openness by controlling the TIA Portal externally from a program you have created.

You can perform the following actions with TIA Portal Openness:

- Create project data
- Modify projects and project data
- Delete project data
- Read in project data
- Make projects and project data available for other applications.

Note

Siemens is not liable for and does not guarantee the compatibility of the data and information transported via these interfaces with third-party software.

We expressly point out that improper use of the interfaces can result in data loss or production downtimes.

Note

The code snippets contained in this documentation are written in C# syntax.

Due to the shortness of the used code snippets the error handling description has been shortened as well.

Application

The TIA Portal Openness interface is used to do the following:

- Provide project data.
- Access the TIA Portal process.
- Use project data.

Using defaults from the field of automation engineering

- By importing externally generated data
- By remote control of the TIA Portal for generating projects

Providing project data of the TIA Portal for external applications

- By exporting project data

Ensuring competitive advantages through efficient engineering

- You do not have to configure existing engineering data in the TIA Portal.
- Automated engineering processes replace manual engineering.
- Low engineering costs strengthen the bidding position as compared to the competition.

Working together on project data

- Test routines and bulk data processing can take place in parallel to the ongoing configuration.

See also

Configurations (Page 46)

4.2.2 Applications

Introduction

TIA Portal Openness provides you with various ways to access the TIA Portal and offers a selection of functions for defined tasks.

You access the following areas of the TIA Portal by using the TIA Portal Openness API interface :

- Portal data
- Project data
- PLC data
- HMI data
- Drive data

Accessing the TIA Portal

TIA Portal Openness offers various ways to access the TIA Portal. You create an external TIA Portal instance in the process either with or without UI. You can also access ongoing TIA Portal processes at the same time.

Accessing projects and project data

When accessing projects and project data, you mainly use TIA Portal Openness for the following tasks:

- Close, open and save the project
- Enumerate and query objects
- Create objects
- Delete objects

4.2.3 Export/import

Introduction

TIA Portal Openness supports the import and export of project data by means of XML files. The import/export function supports external configuration of existing engineering data. You use this function to make the engineering process effective and free of error.

Application

You use the import/export function for the following purposes:

- Data exchange
- Copying parts of a project
- External processing of configuration data, for example, for bulk data operations using find and replace
- External processing of configuration data for new projects based on existing configurations
- Importing externally-created configuration data, for example, text lists and tags
- Providing project data for external applications

TIA Portal Openness API

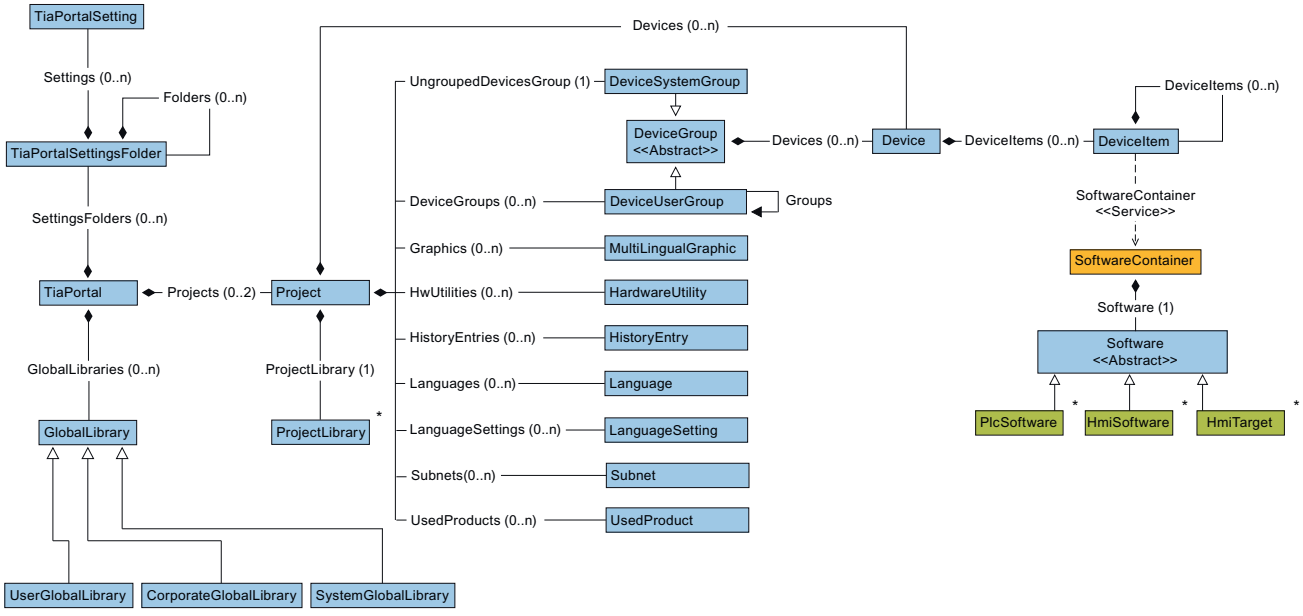
5

5.1 TIA Portal Openness object

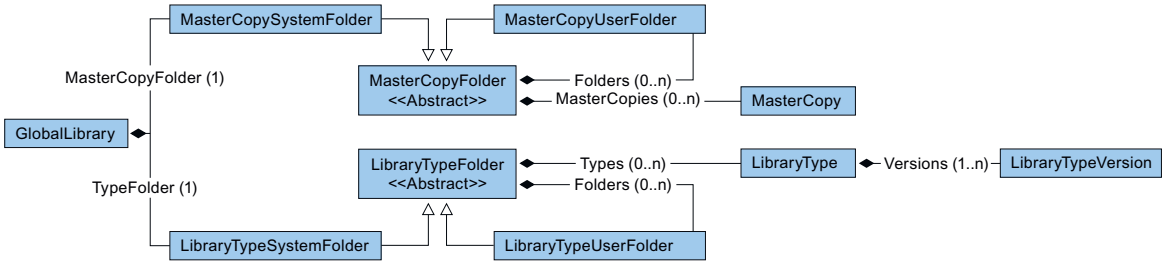
5.1.1 TIA Portal Openness object model

Overview

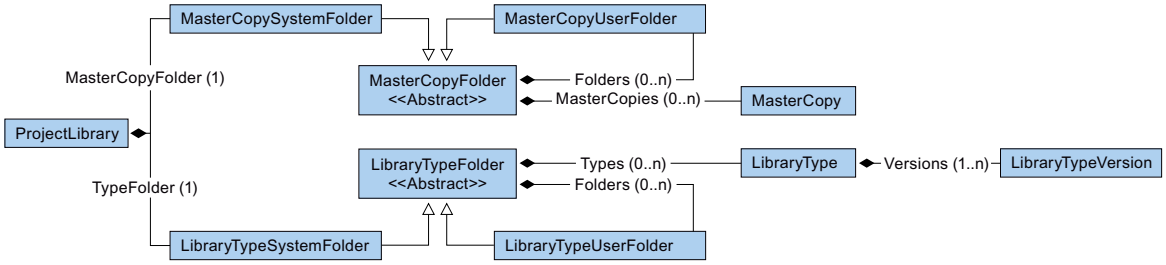
The following diagram describes the highest level of the TIA Portal Openness object model:



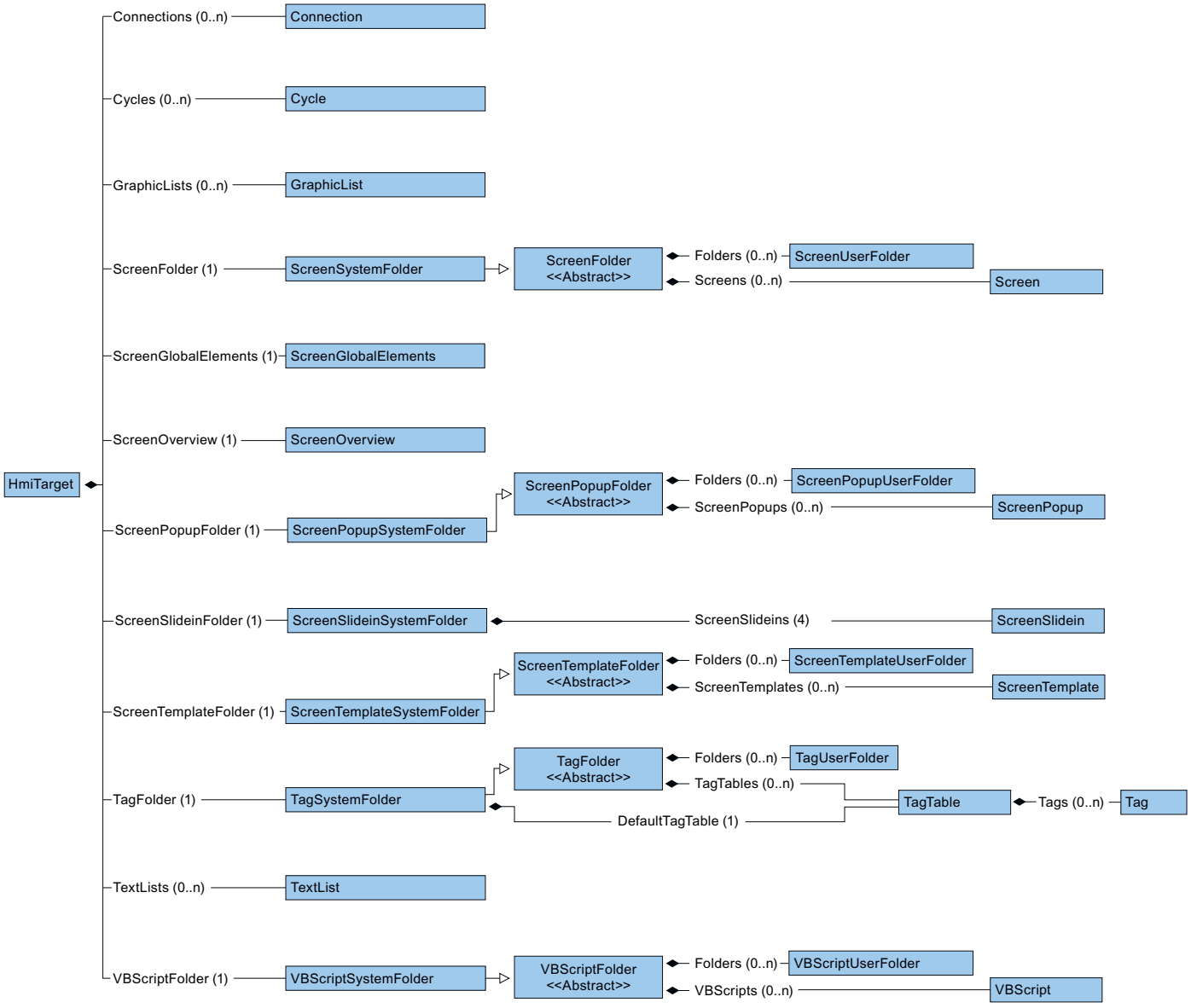
The following diagram describes the objects which are located under GlobalLibrary.



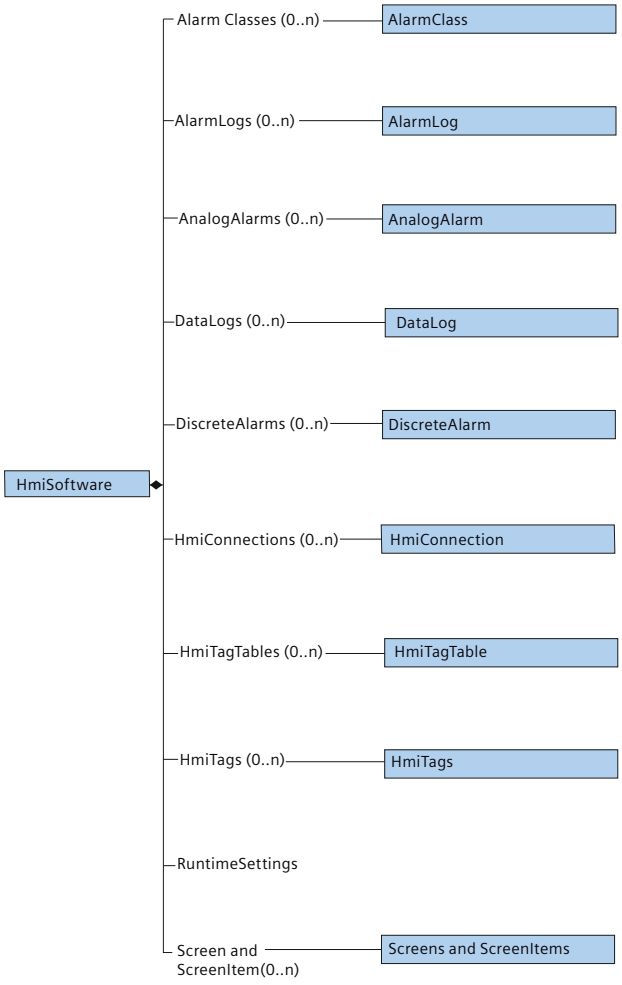
The following diagram describes the objects which are located under ProjectLibrary.



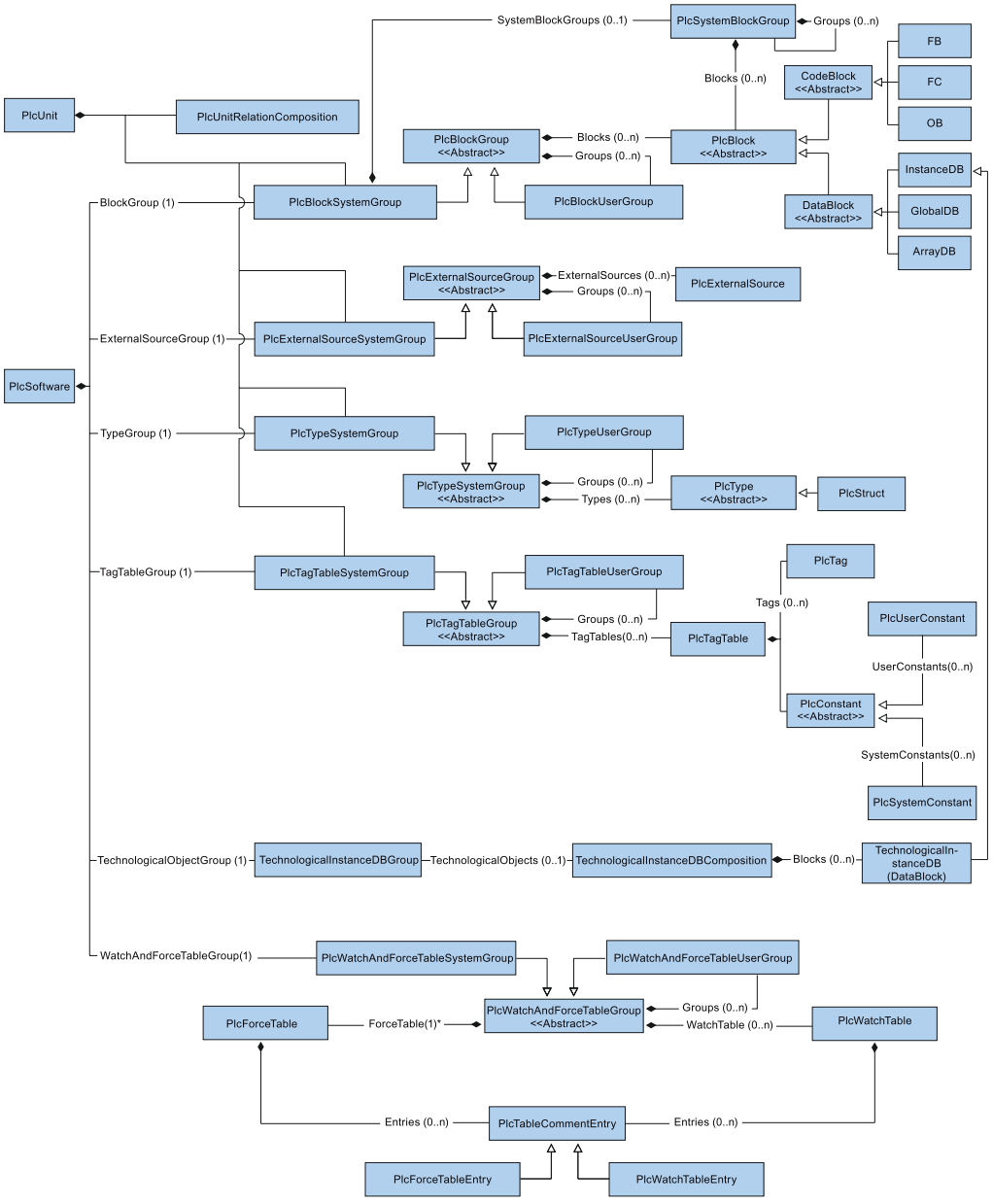
The following diagram describes the objects which are located under HmiTarget.



The following diagram describes the object which are located under HmiSoftware.



The following diagram describes the objects which are located under PlcSoftware.



Note

*The Force Table must be in PlcWatch and ForceTableSystemGroup

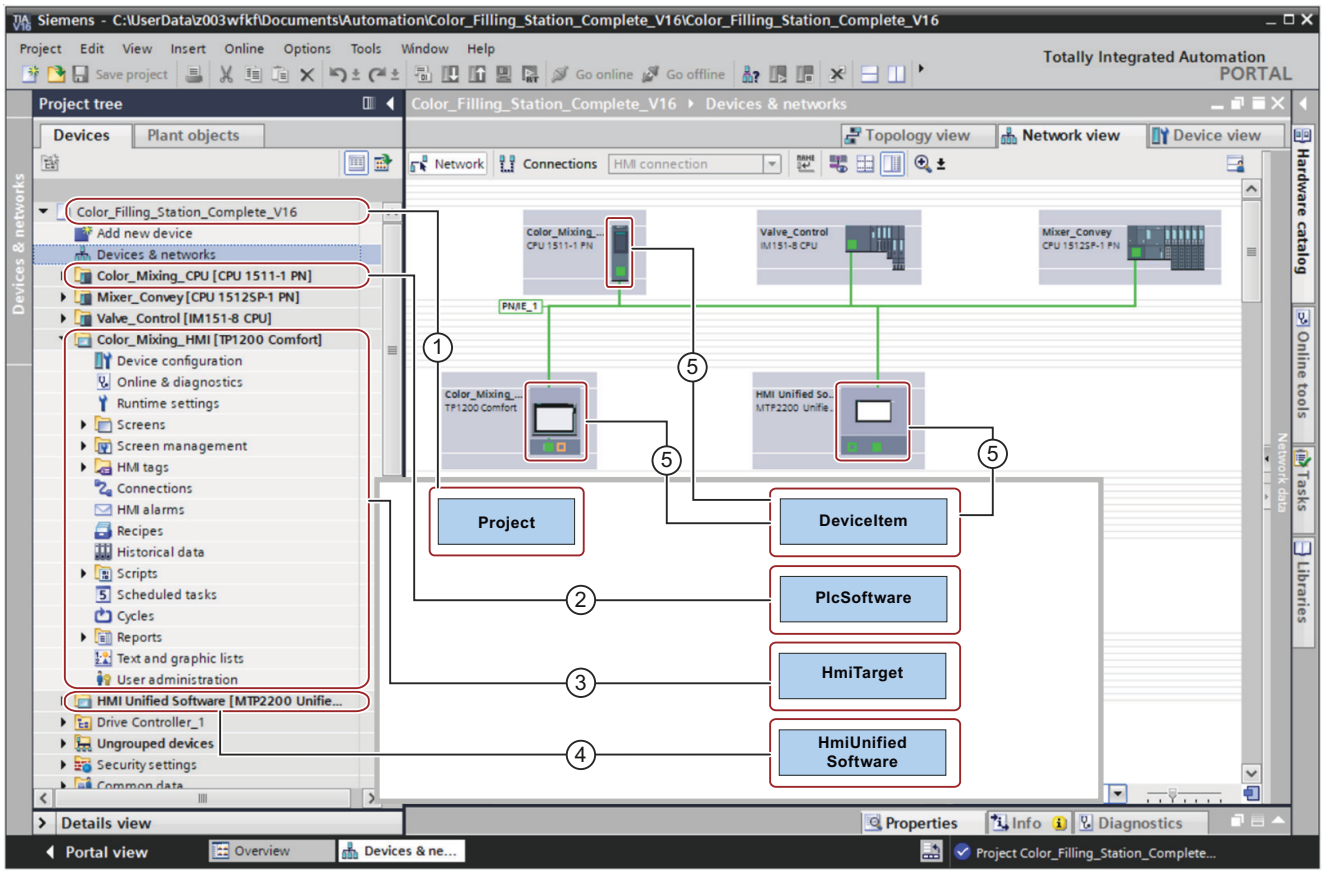
Access to objects in lists

You have the following options for addressing an object in a list:

- Address via the index. The counting within the lists starts with `0`.
- Use `Find` method.
  Use this method to address an object via its name. You can use this method for an composition or list. The `Find` method is not recursive.
  Example:
  ```
  ScreenComposition screens = folder.Screens;
  Screen screen = screens.Find("myScreen");
  ```
- Use symbolic names.

Relationship between TIA Portal and TIA Portal Openness object model

The figure below shows the relationship between the object model and a project in the TIA Portal:



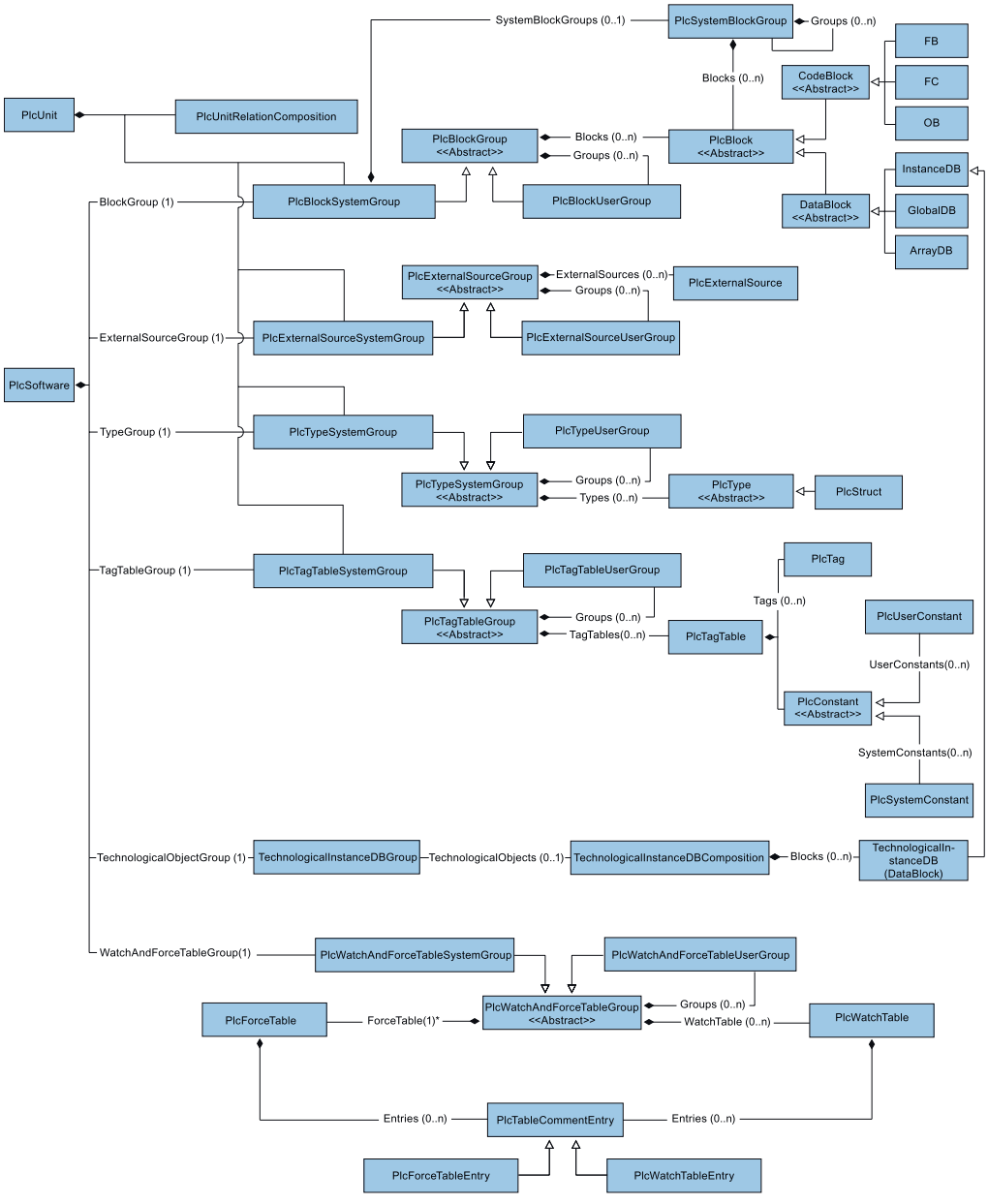
See also

Blocks and types of the TIA Portal Openness object model (Page 59)
Hierarchy of hardware objects of the object model (Page 65)

5.1.2 Blocks and types of the TIA Portal Openness object model

Introduction

The following diagram describes the domain model of the PLCs to give an overview of the current modeling in TIA Portal Openness.



Note

* The Force Table must be in PlcWatch and ForceTableSystemGroup

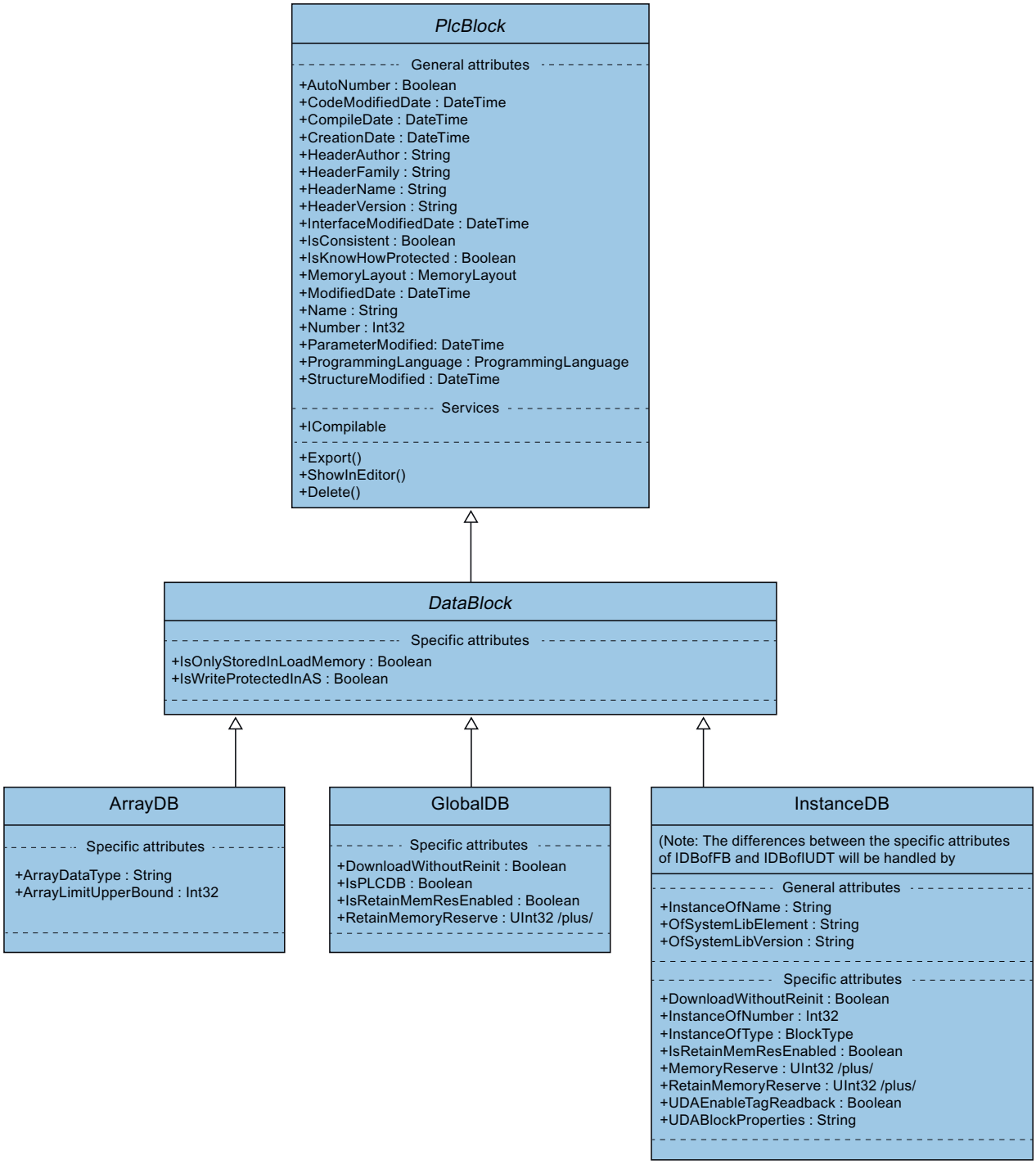
Representation of blocks and types in the TIA Portal Openness API

The simplified model part of blocks and for the structure is based on the attributes in the TIA Portal Openness API. These classes provide the export function and for blocks also the compile function.

Class diagrams

In the TIA Portal Openness object model all classes are defined as abstract which aren't directly instantiated.

Data


PlcBlock
General attributes
+AutoNumber : Boolean
+CodeModifiedDate : DateTime
+CompileDate : DateTime
+CreationDate : DateTime
+HeaderAuthor : String
+HeaderFamily : String
+HeaderName : String
+HeaderVersion : String
+InterfaceModifiedDate : DateTime
+IsConsistent : Boolean
+IsKnowHowProtected : Boolean
+MemoryLayout : MemoryLayout
+ModifiedDate : DateTime
+Name : String
+Number : Int32
+ParameterModified: DateTime
+ProgrammingLanguage : ProgrammingLanguage
+StructureModified : DateTime
Services
+ICompilable
+Export()
+ShowInEditor()
+Delete()
DataBlock
Specific attributes
+IsOnlyStoredInLoadMemory : Boolean
+IsWriteProtectedInAS : Boolean
ArrayDB
Specific attributes
+ArrayDataType : String
+ArrayLimitUpperBound : Int32
GlobalDB
Specific attributes
+DownloadWithoutReinit : Boolean
+IsPLCDB : Boolean
+IsRetainMemResEnabled : Boolean
+RetainMemoryReserve : UInt32 /plus/
InstanceDB
(Note: The differences between the specific attributes of IDBofFB and IDBofIUDT will be handled by
General attributes
+InstanceOfName : String
+OfSystemLibElement : String
+OfSystemLibVersion : String
Specific attributes
+DownloadWithoutReinit : Boolean
+InstanceOfNumber : Int32
+InstanceOfType : BlockType
+IsRetainMemResEnabled : Boolean
+MemoryReserve : UInt32 /plus/
+RetainMemoryReserve : UInt32 /plus/
+UDAEnableTagReadback : Boolean
+UDABlockProperties : String

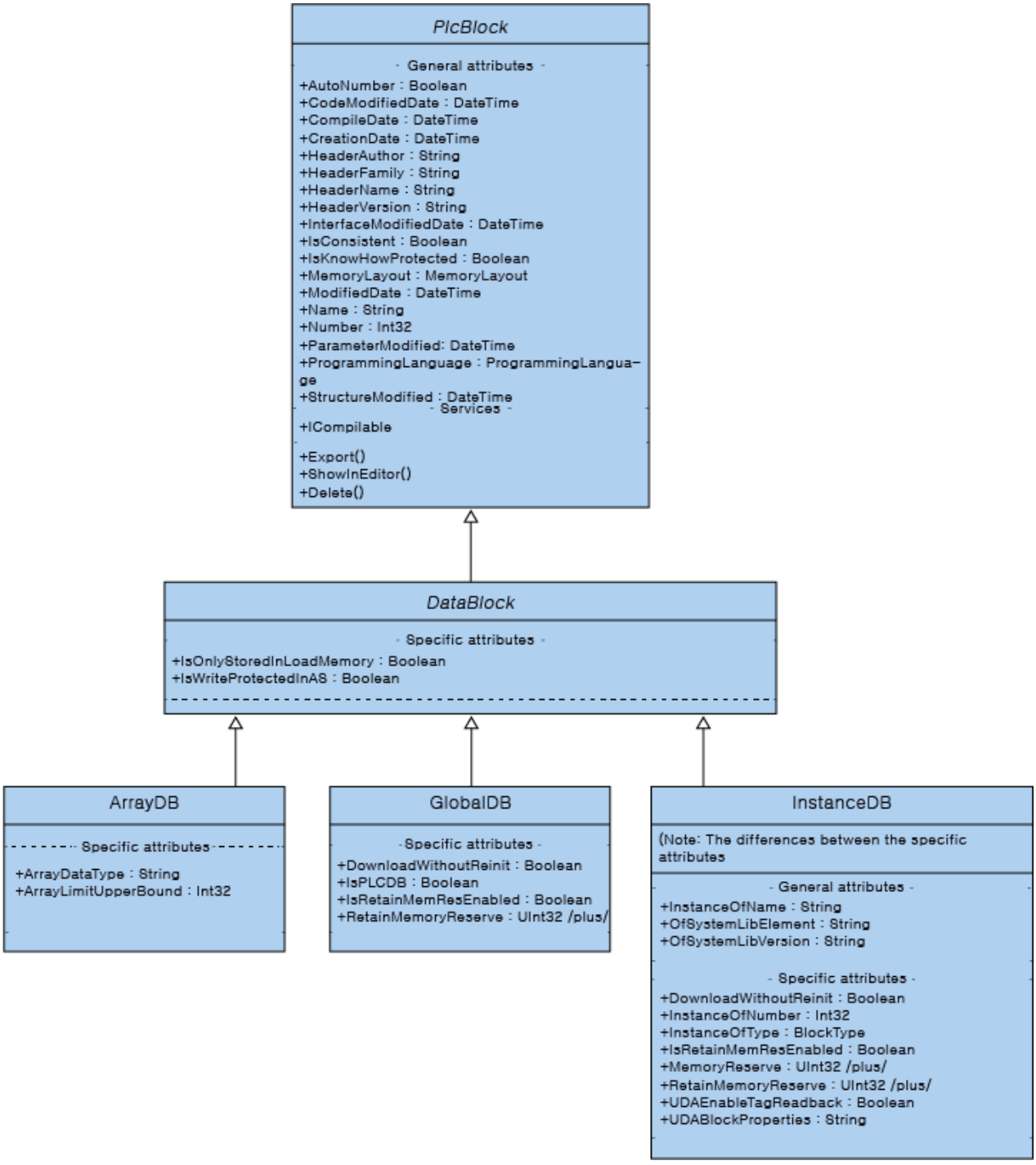
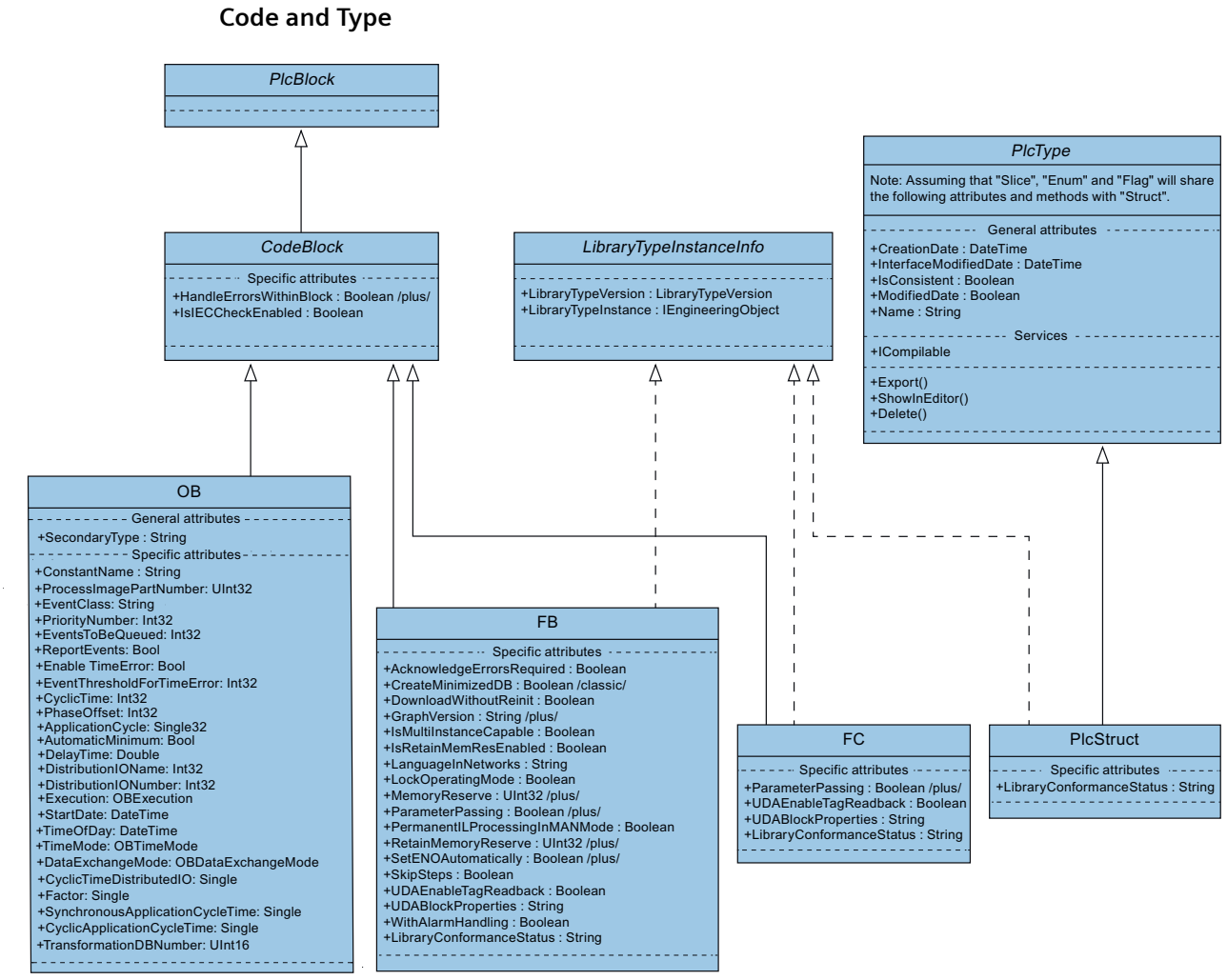


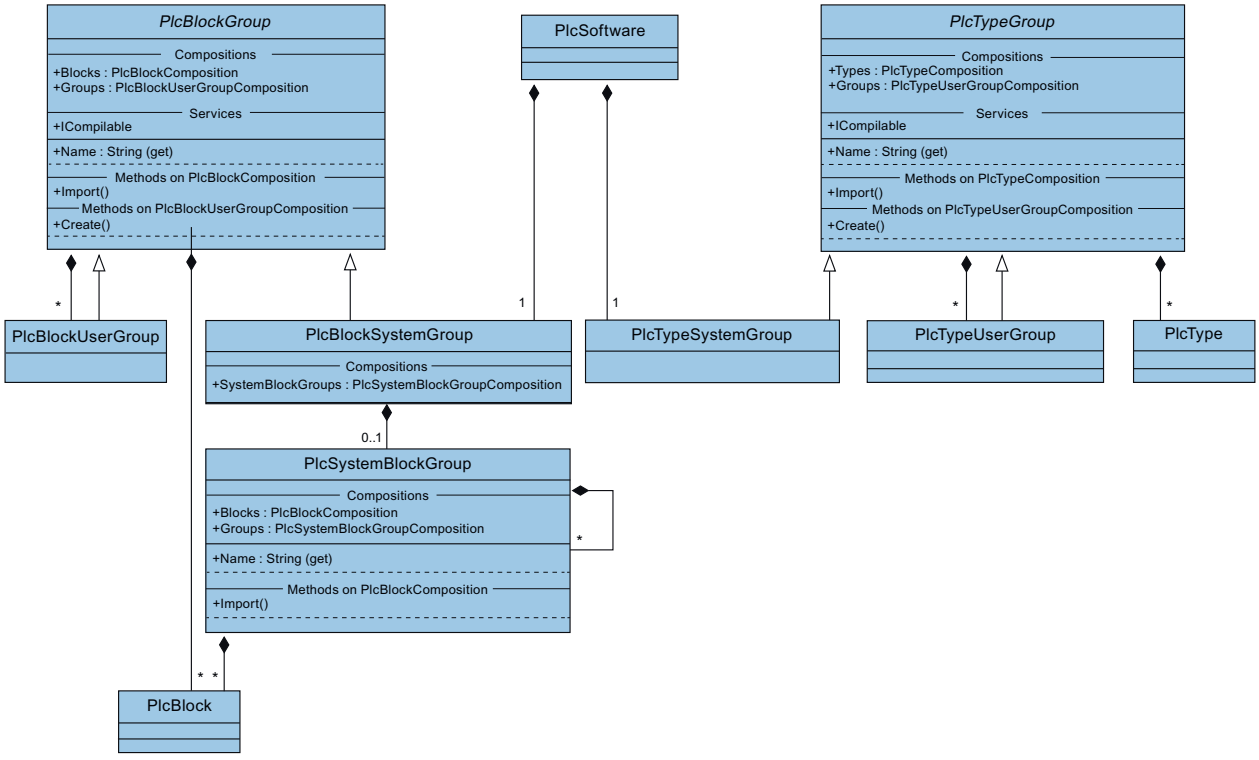
Figure 5-1 Representation of block diagram



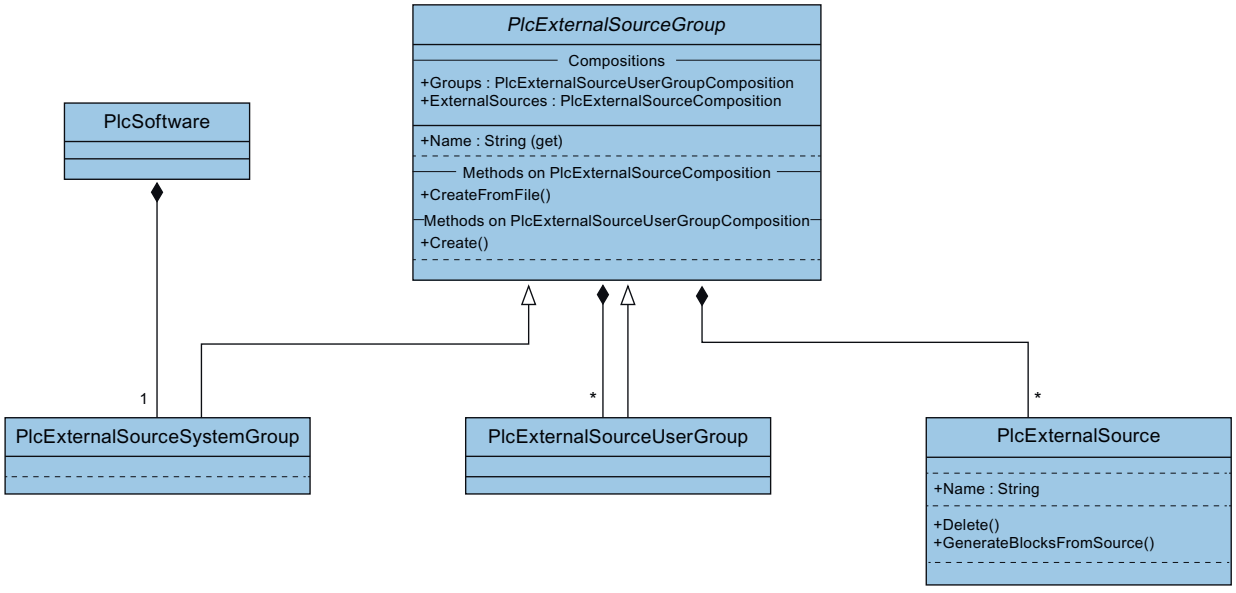
Representation of groups for blocks and types in the TIA Portal Openness API

The two high level groups "PlcBlocks" ("Program blocks" in the GUI of TIA Portal) and "PlcTypes" ("Plc data types" in the GUI of TIA Portal) contain blocks and type definitions. These groups provide the import and the compile functions for blocks. Due to the fact that most of the methods of functionalities of the groups are only achievable via collections, there is an "embedded" or "compacted" representation of the collections and their methods at the "host" classes.

Blocks and Types



External Sources



See also

TIA Portal Openness object model (Page 53)

Hierarchy of hardware objects of the object model (Page 65)

5.1.3 Hierarchy of hardware objects of the object model

Relation between the visible elements in the TIA Portal and the modeled elements in the object model

Hardware object	Explanation
Device (Device)	The container object for a central or distributed configuration.
Device item (DeviceItem)	Each device item object has a container object. The logical relation is "Items".

To understand the hierarchy relationship of hardware objects between what is displayed in the TIA Portal view and what is modeled in the TIA Portal Openness, it is necessary to consider some aspects of the hierarchy of hardware objects.

Example: A HardwareObject has the items link. The Device extends the HardwareObject and has the UnpluggedItems link. The Device is the container object for a station.

The DeviceItem extends the HardwareObject and has the Container link. Every plugged DeviceItem object has got a container object which is a HardwareObject. The reverse relation is Items.

An unplugged DeviceItem object that does not have a container is found via the link UnpluggedItems of the Device.

The container relation is comparable to the relation of the modules for the device item objects.

Example: A device includes one or more racks. A rack includes modules. A module includes submodules.

This is the relation similar to the representation in the network view and device view of the TIA Portal. The "PositionNumber" attribute of a device item is unique in the items area, within a container.

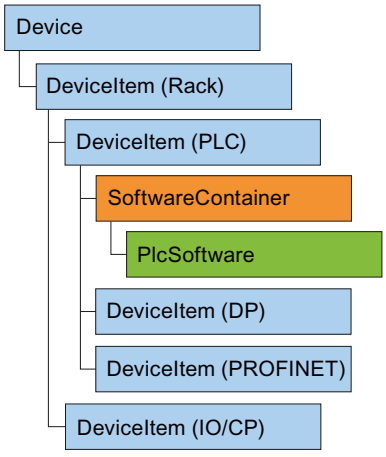
The parent-child relation between device item objects is a purely logical relation in the object model. A child cannot exist without its parents.

- If a submodule is modeled as part of a module (child), the submodule cannot be removed without the module.
- A module or submodule that can be added and removed is a child of the device object

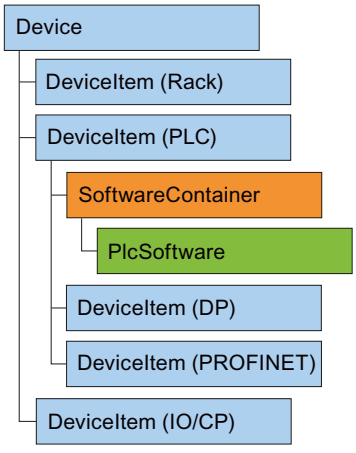
Hierarchy relationships of PLC devices

The diagrams below show the hierarchy relationship between devices, and device items of PLC and HMI devices.

Hierarchy when iterating via .Items

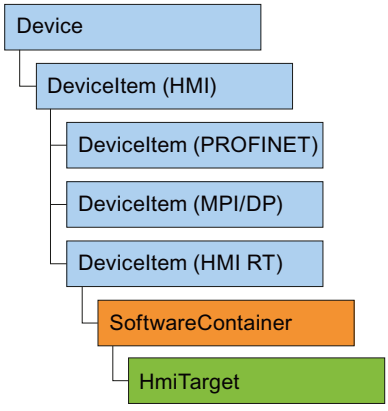


Hierarchy when iterating via .DeviceItems

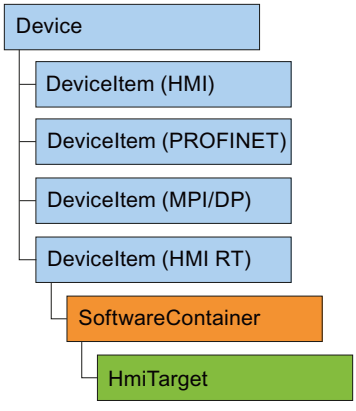


Hierarchy relationships of HMI devices

Hierarchy when iterating via .Items



Hierarchy when iterating via .DeviceItems



See also

TIA Portal Openness object model (Page 53)

Blocks and types of the TIA Portal Openness object model (Page 59)

5.1.4 Object list

Introduction

The following tables show the available objects up to and including Runtime Advanced, and indicate whether these objects are supported by TIA Portal Openness.

Neither Runtime Professional nor device proxy files are supported by TIA Portal Openness in WinCC.

Objects

You can control the following project data depending on which HMI device you are using:

Table 5-1 Screens

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Screen	Yes	Yes	Yes	Yes
Global screen	Yes	Yes	Yes	Yes
Templates	Yes	Yes	Yes	Yes
Permanent area	No	Yes	Yes	Yes
Pop-up screen	No	Yes	Yes	Yes
Slide-in screen	No	Yes	Yes	Yes

Table 5-2 Screen objects

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Line	Yes	Yes	Yes	Yes
Polyline	No	Yes	Yes	Yes
Polygon	No	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes
Ellipse segment	No	No	No	No
Circle segment	No	No	No	No
Elliptical arc	No	No	No	No
Camera view	No	No	No	No
Circular arc	No	No	No	No
Circle	Yes	Yes	Yes	Yes
PDF view	No	No	No	No
Rectangle	Yes	Yes	Yes	Yes
Connector	No	No	No	No
Text field	Yes	Yes	Yes	Yes
Graphic view	Yes	Yes	Yes	Yes
Pipe	No	No	No	No
Double T-piece	No	No	No	No

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
T-piece	No	No	No	No
Pipe bends	No	No	No	No
I/O field	Yes	Yes	Yes	Yes
Date/time field	Yes	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes	Yes
Editable text field	No	No	No	No
List box	No	No	No	No
Combo box	No	No	No	No
Button	Yes	Yes	Yes	Yes
Round button	No	No	No	No
Illuminated button	No	No	Yes	No
Switch	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes
Key-operated switch	No	No	Yes	No
Bar	Yes	Yes	Yes	Yes
Symbol library	No	Yes	Yes	Yes
Slider	No	Yes	Yes	Yes
Scroll bar	No	No	No	No
Check box	No	No	No	No
Option buttons	No	No	No	No
Gauge	No	Yes	Yes	Yes
Clock	No	Yes	Yes	Yes
Memory space view	No	No	No	No
Function keys	Yes	Yes	Yes	Yes
Faceplate instances	No	Yes	Yes	Yes
Screen window	No	No	No	No
User view	Yes	Yes	Yes	Yes
HTML Browser	No	No	No	No
Print job/script diagnostics	No	No	No	No
Recipe view	No	No	No	No
Alarm view	No	No	No	No
Alarm indicator	No	No	No	No
Alarm window	No	No	No	No
Criteria analysis view	No	Yes [1)]	Yes	Yes
ProDiag overview	No	Yes [1)]	Yes	Yes
GRAPH overview	No	Yes [1)]	Yes	Yes
PLC code view	No	Yes [1)]	Yes	Yes
f(x) trend view	No	No	No	No
f(t) trend view	No	No	No	No
Table view	No	No	No	No

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Value table	No	No	No	No
Media Player	No	No	No	No
Channel diagnostics	No	No	No	No
WLAN reception	No	No	No	No
Zone name	No	No	No	No
Zone signal	No	No	No	No
Effective range name	No	No	No	No
Effective range name (RFID)	No	No	No	No
Effective range signal	No	No	No	No
Charge condition	No	No	No	No
Handwheel	No	No	Yes	No
Help indicator	No	No	No	No
Sm@rtClient view	No	No	No	No
Status/Force	No	No	No	No
System diagnostic view	No	No	No	No
System diagnostic window	No	No	No	No

1) Only Mobile Panels with the device version greater than 12.0.0.0 support this screen object

Table 5-3 Dynamic

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Display	Yes	Yes	Yes	Yes
Operability	No	Yes	Yes	Yes
Visibility	Yes	Yes	Yes	Yes
Movements	Yes	Yes	Yes	Yes

Table 5-4 Additional objects

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Groups	Yes	Yes	Yes	Yes
Soft keys	Yes	Yes	Yes	Yes
Cycles	Yes	Yes	Yes	Yes
VB scripts	No	Yes	Yes	Yes
Function lists	Yes	Yes	Yes	Yes
Project graphics	Yes	Yes	Yes	Yes

Table 5-5 Tags

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Multiplex tags	Yes	Yes	Yes	Yes
Arrays	Yes	Yes	Yes	Yes
User data types	Yes	Yes	Yes	Yes
Internal	No	Yes	Yes	Yes
Points of use of elementary data types	Yes	Yes	Yes	Yes
Points of use of user data types	Yes	Yes	Yes	Yes
Points of use of arrays	Yes	Yes	Yes	Yes

TIA Portal Openness also supports all value ranges which are supported by the communication drivers.

Connections

TIA Portal Openness supports non-integrated connections that are also supported by the respective HMI devices. You can find additional information in the online help for the TIA Portal under "Process visualization > Controller communication > Device-dependent".

Table 5-6 Lists

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Text lists	Yes	Yes	Yes	Yes
Graphic lists	Yes	Yes	Yes	Yes

Table 5-7 Texts

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Multilingual texts	Yes	Yes	Yes	Yes
Formatted texts and their instances	No	Yes	Yes	Yes

5.2 General functions

5.2.1 Introduction

Overview

TIA Portal Openness supports a selection of functions for defined tasks that you can call outside the TIA Portal by means of the TIA Portal Openness API.

Note

If a previous version of TIA Portal Openness is already installed, the current version will be installed side by side.

You are provided with an overview of the typical programming steps in the sections below. You can learn how the individual code sections interact and how to integrate the respective functions into a complete program. You also get an overview of the code components that have to be adapted for each task.

Example program

The individual programming steps are explained using the "Creating API access in a console application" function as an example. You integrate the provided functions in this program code and adapt the respective code components for this task.

Functions

The section below lists the functions for defined tasks that you can call with TIA Portal Openness outside the TIA Portal.

See also

Applications (Page 50)

Object list (Page 67)

5.2.2 Information about installed TIA Portal Openness versions

Requirement

- TIA Portal Openness and TIA Portal are installed

Application

Starting from TIA Portal Openness V14 each installed version has a registry key that contains information about the version. This enables an automatic generation of app.config files for each installed version of TIA Portal Openness.

The registry keys can be located under the following path:

```
HKEY_LOCAL_MACHINE\Software\Siemens\Automation\Openness\xx.x\PublicAPI
```

Note

The version number in this path, is always the number of the currently installed version of TIA Portal. If there are multiple side-by-side installations there are multiple sets of entries for TIA Portal Openness in the registry.

There is a single key for each version of TIA Portal Openness. The names of the Versions will be the same as in the assembly described, for example, the registry entries for TIA Portal Openness:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\xx.x\PublicAPI\xx.x.x.x]"PublicKeyToken"="d29ec89bac048f84"
"Siemens.Engineering"="C:\Program Files\Siemens\Automation\Portal Vxx\PublicAPI\Vxx\Siemens.Engineering.dll"
"Siemens.Engineering.Hmi"="C:\Program Files\Siemens\Automation\Portal Vxx\PublicAPI\Vxx\Siemens.Engineering.Hmi.dll"
"EngineeringVersion"="xx"
"AssemblyVersion"="xx.x.x.x"
```

Note

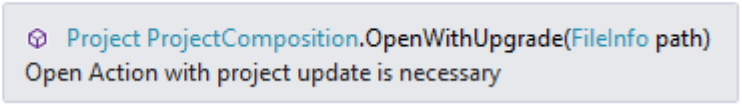
If you want to generate an app.config file (Page 82) you can get the path of the Siemens.Engineering.dll, the Siemens.Engineering.Hmi.dll and the public key token from the registry key.

5.2.3 TIA Portal Openness IntelliSense support

Application

The Intellisense support of TIA Portal Openness helps you at available attributes or methods via tooltip information. It could contain information about the number, names and types of the required parameters. At the following example the bold parameter in the first line indicates the next parameter that is required as you type the function.

```
project = tiaPortal.Projects.OpenWithUpgrade(projectInfo);
```



You can manually invoke Parameter Info by clicking Edit IntelliSense/Parameter Info, typing CTRL+SHIFT+SPACE, or clicking the Parameter Info button on the editor toolbar.

5.2.4 Standard libraries

Insert the following namespace statements at the beginning of the relevant code example to ensure that the code examples work:

```
using System;
using System.Collections.Generic;
using System.IO;
using Siemens.Engineering;
using Siemens.Engineering.Cax;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Compare;
using Siemens.Engineering.Download;
using Siemens.Engineering.Hmi;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.RuntimeScripting;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Extensions;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.HW.Utilities;
using Siemens.Engineering.Library;
using Siemens.Engineering.Library.MasterCopies;
using Siemens.Engineering.Library.Types;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.TechnologicalObjects;
using Siemens.Engineering.SW.TechnologicalObjects.Motion;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Upload;
```

5.2.5 Use of the code examples

Structure of the code-snippets

Each code-snippet in this documentation is implemented as a function without return value with an object reference as transfer parameter. Disposing of objects is omitted for the sake of readability. Objects of the TIA Portal are addressed by their name using the `Find` method.

```
//Deletes a single screen from a user folder or a system folder
private static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    //The screen "MyScreen" will be deleted if it is existing in the folder
"myScreenFolder".
    //If "myScreen" is stored in a subfolder of "myScreenFolder" it will not be deleted.
    string screenName = "MyScreen";
    ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Delete();
    }
}
```

You need the following to execute this code-snippet:

- A WinCC project with an HMI device that includes a group with at least one screen.
- A function that instantiates the HMI device.

Note

When you specify directory paths, use the absolute directory path, for example, "C:/path/file.txt".

Relative directory paths are only allowed in the XML files for import and export, for example, "file.txt" or "C:/path01/.../path02/file.txt".

Example for execution of the code-snippet

Use the following example to execute the code-snippet "DeleteScreenFromFolder" as part of the "Hello TIA" example program:

```
//In the sample program "Hello TIA" replace the function call
//"IterateThroughDevices(project)" by the following functions calls:
    HmiTarget hmiTarget = GetTheFirstHmiTarget(project);
    DeleteScreenFromFolder(hmiTarget);

//Put the following function definitions before or after the
//function definition of "private static void IterateThroughDevices(Project project)":
private static HmiTarget GetTheFirstHmiTarget(Project project)
{
    if (project == null)
    {
        Console.WriteLine("Project cannot be null");
        throw new ArgumentNullException("project");
    }
    foreach (Device device in project.Devices)
    //This example looks for devices located directly in the project.
    //Devices which are stored in a subfolder of the project will not be affected by this
example.
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            DeviceItem deviceItemToGetService = deviceItem as DeviceItem;
            SoftwareContainer container =
deviceItemToGetService.GetService<SoftwareContainer>();
            if (container != null)
            {
                HmiTarget hmi = container.Software as HmiTarget;
                if (hmi != null)
                {
                    return hmi;
                }
            }
        }
    }
    return null;
}

//Deletes a single screen from a user folder or a system folder
private static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    string screenName = "MyScreen";
    ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Delete();
    }
}
```

5.2.6 Example program

Application example: Creating API access in an application

The complete program code of the application example is shown below. The typical programming steps are explained next based on this example.

Note

The application example requires an application configuration file (Page 82).

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
namespace HelloTIA
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            RunTiaPortal();
        }

        private static void RunTiaPortal()
        {
            Console.WriteLine("Starting TIA Portal");
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                Console.WriteLine("TIA Portal has started");
                ProjectComposition projects = tiaPortal.Projects;

                Console.WriteLine("Opening Project...");
                // please adapt the path and the extension apx to the installed version of
TIA Portal
                FileInfo projectPath = new FileInfo("C:\\Demo\\AnyCompanyProject.apx"); //
edit the path according to your project
                Project project = null;
                try
                {
                    project = projects.OpenWithUpgrade(projectPath);
                }
                catch (Exception)
                {
                    Console.WriteLine(String.Format("Could not open project {0}",
projectPath.FullName));
                    Console.WriteLine("Demo complete hit enter to exit");
                    Console.ReadLine();
```

```
                    return;
                }

                Console.WriteLine(String.Format("Project {0} is open",
project.Path.FullName));

                IterateThroughDevices(project);

                project.Close();

                Console.WriteLine("Demo complete hit enter to exit");
                Console.ReadLine();
            }
        }

        private static void IterateThroughDevices(Project project)
        {
            if (project == null)
            {
                Console.WriteLine("Project cannot be null");
                return;
            }

            Console.WriteLine(String.Format("Iterate through {0} device(s)",
project.Devices.Count));

            foreach (Device device in project.Devices)
            {
                Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
            }

            Console.WriteLine();
        }
    }
}
```

Procedure in steps

1. Make the TIA Portal known in the development environment

In your development environment, create a reference to all "dll files" in the "C:\Program Files\Siemens\Automation\PortalV..\PublicAPI\V.." directory.

The following provides a description of this process using the "Siemens.Engineering.dll" file as an example.

The "Siemens.Engineering.dll" file is available in the directory "C:\Program Files\Siemens\Automation\PortalV..\PublicAPI\V..". Create a reference to the "Siemens.Engineering.dll" file in your development environment.

Note

Ensure that parameter "CopyLocal" is assigned the value "False" in the reference attributes.

2. Publish the name space for the TIA Portal

Add the following code:

```
using Siemens.Engineering;
```

3. Publish and start the TIA Portal

In order to publish and start the TIA Portal, insert the following code:

```
using (TiaPortal tiaPortal = new TiaPortal())
{
    // Add your code here
}
```

4. Open project

You can use the following code, for example, to open a project:

```
ProjectComposition projects = tiaPortal.Projects;
Console.WriteLine("Opening Project...");
//please adapt the path and the extension apx to the installed version of TIA Portal
FileInfo projectPath = new FileInfo("C:\Demo\AnyCompanyProject.apx");
Project project = null;
try
{
    project = projects.Open(projectPath);
}
catch (Exception)
{
    Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
    Console.WriteLine("Demo complete hit enter to exit");
    Console.ReadLine();
    return;
}
Console.WriteLine(String.Format("Project {0} is open", project.Path.FullName));
```

5. Enumerate devices of a project

Insert the following code to enumerate all devices of the project:

```
static private void IterateThroughDevices(Project project)
    {
        if (project == null)
        {
            Console.WriteLine("Project cannot be null");
            return;
        }

        Console.WriteLine();
        Console.WriteLine(String.Format("Iterate through {0} device(s)",
project.Devices.Count));
        foreach (Device device in project.Devices)
        {
            Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
        }
        Console.WriteLine();
    }
```

6. Save and close the project

Insert the following code to save and close the project:

```
project.Save();
project.Close();
```

5.2.7 Programming steps

Overview

TIA Portal Openness requires the following programming steps for access by means of the TIA Portal Openness API:

1. Make the TIA Portal known in the development environment
2. Set up program access to the TIA Portal
3. Activate program access to the TIA Portal
4. Publish and start the TIA Portal
5. Open project
6. Execute commands
7. Save and close the project
8. Terminate the connection to the TIA Portal

Note

Permitted strings

Only certain characters are allowed in strings in the TIA Portal. All strings passed to the TIA Portal via the TIA Portal Openness application are subject to these rules. If you pass an invalid character in a parameter, an exception is thrown.

See also

Example program (Page 77)

Use of the code examples (Page 75)

5.2.8 Connecting to the TIA Portal

Introduction

You start the TIA Portal with TIA Portal Openness or connect to a TIA Portal already running. When using a TIA Portal Openness application to start the TIA Portal, you specify if the TIA Portal should be started with or without graphical user interface. When you operate the TIA Portal without user interface, the TIA Portal is only started as a process by the operating system. You create several instances of the TIA Portal with a TIA Portal Openness application, if necessary.

Note

If you use TIA Portal Openness with the TIA Portal interface, you cannot use an HMI editor. You can open the "Devices & Networks" editors or the programming editor manually or with TIA Portal Openness API.

You have the following options to start the TIA Portal with a TIA Portal Openness application:

- Use an application configuration file (recommended in most use cases).
- Use the "AssemblyResolve" method (recommended when you use copy deploy etc.).
- Copy the Siemens.Engineering.dll in the TIA Portal Openness application directory.

Note

It is recommended to load the Siemens.Engineering.dll by using the app.config file. By using this method the strong names are considered and malicious modifications to the engineering.dll will result in a loading error. By using the AssemblyResolve method this can't be detected.

Starting the TIA Portal with an application configuration file

Reference all required program libraries in the application configuration file. You distribute the application configuration file together with the TIA Portal Openness application.

Store the application configuration file "app.config" in the same directory as the TIA Portal Openness application and likewise incorporate this in your application. Check whether the file path in each code matches the TIA Portal installation path.

You can use the following code snippet for the application configuration file:

```
<?xml version="1.0"?>
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
            <dependentAssembly>
                <assemblyIdentity name="Siemens.Engineering" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
             <!-- Edit the following path according to your installed version of TIA Portal
-->
                <codeBase version="xx.x.x.x" href="FILE://C:\Program
Files\Siemens\Automation\Portal Vxx\PublicAPI\Vxx\Siemens.Engineering.dll"/>
            </dependentAssembly>
            <dependentAssembly>
                <assemblyIdentity name="Siemens.Engineering.Hmi" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
             <!-- Edit the following path according to your installed version of TIA Portal
-->
                <codeBase version="xx.x.x.x" href="FILE://C:\Program
Files\Siemens\Automation\Portal Vxx\PublicAPI\Vxx\Siemens.Engineering.Hmi.dll"/>
            </dependentAssembly>
        </assemblyBinding>
    </runtime>
</configuration>
```

Use the following program code to open a new TIA Portal Instance by means of the application configuration file:

```
//Connect a TIA Portal Openness application via API using
using System;
using System.IO;
using Siemens.Engineering;

namespace UserProgram
{
    internal class MyProgram
    {
        public static void Main(string[] args)
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            //
            // To start TIA Portal without user interface:
            // using (TiaPortal tiaPortal = new
TiaPortal(TiaPortalMode.WithoutUserInterface))
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }
    }
}
```

Starting the TIA Portal using the "AssemblyResolve" method

Design the program code of the TIA Portal Openness application in such a way that you register on the event "AssemblyResolve" as early as possible. Encapsulate the access to the TIA Portal in an additional object or method.

Caution must be taken when resolving the engineering assembly using an assembly resolver method. If any types from the engineering assembly are used before the assembly resolver has had run, the program will crash. The reason for this is that the Just-in-time compiler (JIT compiler) doesn't compile methods until it needs to execute them. If engineering assembly types are used in Main, for example, the JIT compiler will attempt to compile Main when the program runs and fail because it doesn't know where to find the engineering assembly. The registration of the assembly resolver in Main doesn't change this. The method needs to run before the assembly resolver is registered, and it needs to be compiled before it can be run. The solution for this problem, is to place the business logic that uses types from the engineering assembly into a separate method that uses only types that the JIT compiler already understands. In the example, a method that returns void and has no parameters and place all business logic inside it is used. When the JIT compiler compiles Main, it will succeed because it knows all the types in Main. At runtime, when we call RunTiaPortal, the assembly resolver will already be registered, so when the JIT compiler tries to find our business logic types, it will know where to find the engineering assembly.

Use the following program code to open a new TIA Portal Instance.

```
using System;
using System.IO;
using System.Reflection;
using Siemens.Engineering;

namespace UserProgram
{
    static class MyProgram
    {
        public static void Main(string[] args)
        {
            AppDomain.CurrentDomain.AssemblyResolve += MyResolver;
            RunTiaPortal();
        }
        private static void RunTiaPortal()
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            //
            // To start TIA Portal without user interface:
            // using (TiaPortal tiaPortal = new
TiaPortal(TiaPortalMode.WithoutUserInterface))
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }
        private static Assembly MyResolver(object sender, ResolveEventArgs args)
        {
            int index = args.Name.IndexOf(',');
            if (index == -1)
            {
                return null;
            }
            string name = args.Name.Substring(0, index) + ".dll";
            // Edit the following path according to your installed version of TIA Portal
            string path = Path.Combine(@"C:\Program Files\Siemens\Automation\Portal
Vxx\PublicAPI\Vxx\", name);
            string fullPath = Path.GetFullPath(path);
            if (File.Exists(fullPath))
            {
                return Assembly.LoadFrom(fullPath);
            }
            return null;
        }
    }
}
```

Accessing running instances of the TIA Portal

In order to connect to a running instance of the TIA Portal with a TIA Portal Openness application, start by enumerating the instances of the TIA Portal. You can connect to multiple instances within a Windows session. The running instance can be TIA Portal with or without a started user interface:

```
foreach (TiaPortalProcess tiaPortalProcess in TiaPortal.GetProcesses())
{
//...
}
```

Note

A TIA Portal instance is not required to enumerate through all running TIA Portal process because GetProcesses is a static method on the class TiaPortal.

If you know the process ID of the instance of the TIA Portal, use this process ID to access the object. TIA Portal requires a certain amount of time to start up before you can connect the TIA Portal Openness application to the TIA Portal.

```
Process process = ...;
Thread.Sleep(TimeNeededToStartUp); // The process may not be ready to service requests
immediately.
TiaPortalProcess tiaPortalProcess = TiaPortal.GetProcess(process.Id);
```

Note

The process id to which user is connected must be in running state and should have reached where it can respond to requests for process information.

When you connect to a running instance of the TIA Portal, a connection prompt of the TIA Portal Openness firewall appears. The connection prompt offers the following options:

- Allow connection once
- Do not allow connection
- Always allow connections from this application
  See TIA Portal Openness firewall (Page 88) for further information.

  Note

  If the registry prompt is rejected three times, the system throws an exception of the type `EngineeringSecurityException`.

Once you have connected to the process, you can use the following attributes to retrieve information on the instances of the TIA Portal:

Attribute	Information
`InstalledSoftware as IList<TiaPortalProduct>`	Returns information about the installed products.
`Mode as TiaPortalMode`	Returns the mode in which the TIA Portal was started (WithoutUserInterface/WithUserInterface).
`AttachedSessions as IList<TiaPortalSession>`	Returns a list of applications connected to the TIA Portal.
`ProjectPath as FileInfo`	Returns the file name of the project opened in the TIA Portal, including the folder, for example, `"D:\WinCCProjects\ColorMixing\ColorMixing.ap*"` If no project is open, a null string is returned.
`ID as int`	Returns the process ID of the TIA Portal instance
`Path as FileInfo`	Returns the path to the TIA Portal executable

Note

TIA Portal Openness does not support a compatibility mode. Each TIA Portal option package or HSP used in the project has to be installed for an Openness application to be able to open or attach to the project.

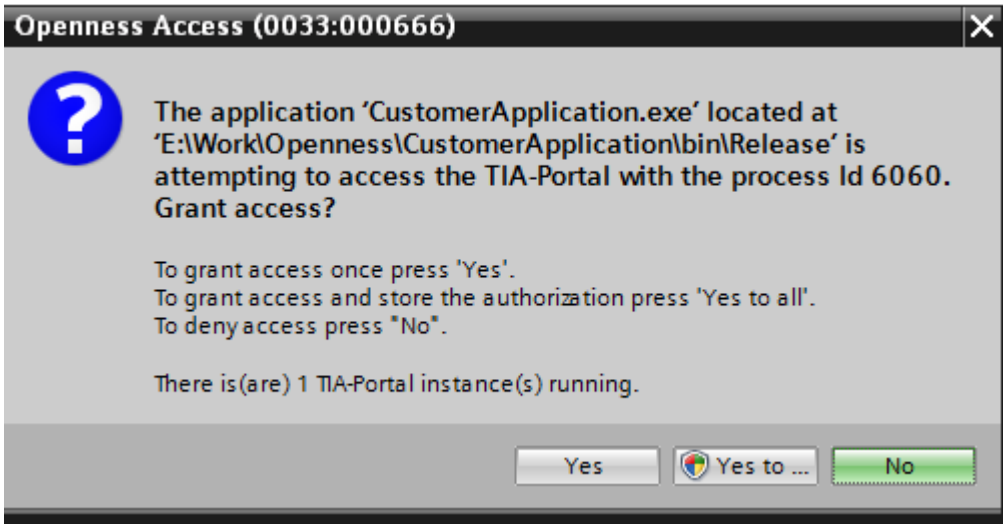
To get the current TIA Portal process from a TIA Portal application by calling GetCurrentProcess() on TIA Portal instance:

```
TiaPortal tiaPortal = ...;
TiaPortalProcess tiaPortalProcess = tiaPortal.GetCurrentProcess();
```

5.2.9 TIA Portal Openness firewall

TIA Portal Openness firewall prompt

When you try to connect to a running TIA Portal via TIA Portal Openness, the TIA Portal will prompt you to accept or reject the connection like the following screenshot is showing.



Allow connection to the TIA Portal once

If you just want to connect your TIA Portal Openness application to the TIA Portal once, click "Yes" at the prompt. The next time your TIA Portal Openness application tries to connect the TIA Portal, the prompt will be shown again.

Addition of a whitelist entry by connecting the TIA Portal

To create a whitlist entry for your TIA Portal Openness application follow these steps:

1. Click "Yes to all" at the prompt to display an User Account Control Dialog.
2. Click "Yes" at the User Account Control Dialog to add your application to the whitelist in the windows registry and to attach the application to the TIA Portal.

Addition of a whitelist entry without using the TIA Portal

If you want to add an entry to the whitelist without using TIA Portal you can create a reg file like this:

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\xx.x\Whitelist\CustomerApplicatio
n.exe]
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\xx.x\Whitelist\CustomerApplicatio
n.exe\Entry]
"Path"="E:\\Work\\Openness\\CustomerApplication\\bin\\Release\\CustomerApplication.exe"
"DateModified"="2014/06/10 15:09:44.406"
"FileHash"="0rXRKUCNzMWHOMFrT52OwXzqJef10ran4UykTeBraaY="
```

The following example shows how you can calculate the file hash and last modified date:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;

using Siemens.Engineering.SW.ExternalSources;

using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security.Cryptography;
namespace TIAPortalOpennessFirewall
{
internal class Program
{
private static void Main(string[] args)
{
RunTiaPortal();

string applicationPath = @"E:\\Work\\Openness\\CustomerApplication\\bin\\Release\
\CustomerApplication.exe";
string lastWriteTimeUtcFormatted = String.Empty;
DateTime lastWriteTimeUtc;
HashAlgorithm hashAlgorithm = SHA256.Create();
FileStream stream = File.OpenRead(applicationPath);
byte[] hash = hashAlgorithm.ComputeHash(stream);
// this is how the hash should appear in the .reg file
string convertedHash = Convert.ToBase64String(hash);
lastWriteTimeUtc = fileInfo.LastWriteTimeUtc; // fileInfo not defined

// this is how the last write time should be formatted
lastWriteTimeUtcFormatted = lastWriteTimeUtc.ToString(@"yyyy/MM/dd HH:mm:ss.fff");


        }
}
private static void RunTiaPortal()
{
Console.WriteLine("Starting TIA Portal");
using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
{
```

```
Console.WriteLine("TIA Portal has started");
ProjectComposition projects = tiaPortal.Projects;
Console.WriteLine("Opening Project...");
// please adapt the path and the extension apx to the installed version of TIA Portal
FileInfo projectPath = new FileInfo(@"C:\Demo\AnyCompanyProject.apx");
//edit the path according to your project
Project project = null;
try
{
project = projects.OpenWithUpgrade(projectPath);
}
catch (Exception)
{
Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
Console.WriteLine("Demo complete hit enter to exit");
Console.ReadLine();
return;
}
Console.WriteLine(String.Format("Project {0} is open", project.Path.FullName));
IterateThroughDevices(project);
project.Close();
Console.WriteLine("Demo complete hit enter to exit");
Console.ReadLine();
}
}
private static void IterateThroughDevices(Project project)
{
if (project == null)
{
Console.WriteLine("Project cannot be null");
return;
}
Console.WriteLine(String.Format("Iterate through {0} device(s)", project.Devices.Count));
foreach (Device device in project.Devices)
{
Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
}
Console.WriteLine();
}
}
}
```

5.2.10 Event handlers

Event handlers in TIA Portal Openness application

An instance of the TIA Portal provides the following events to which you can react with an event handler in a TIA Portal Openness application. You can access the attributes of notifications and define the responses accordingly.

Event	Response
`Disposed`	Use this event to respond to the closing of the TIA Portal with a TIA Portal Openness application.
`Notification`	Use this event to respond to notifications of the TIA Portal with a TIA Portal Openness application. Notifications require only an acknowledgment, e.g. "OK".
`Confirmation`	Use this event to respond to confirmations of the TIA Portal with a TIA Portal Openness application. Confirmations always require a decision, e.g. "Do you want to save the project?".

Program code

Modify the following program code to register event handlers in a TIA Portal Openness application:

```
//Register event handler for Disposed-Event
    ....
        tiaPortal.Disposed +=TiaPortal_Disposed;
....

private static void TiaPortal_Disposed(object sender, EventArgs e)
{
    ....
}

//Register event handler for Notification-Event
    ....
        tiaPortal.Notification += TiaPortal_Notification;
....

private static void TiaPortal_Notification(object sender, NotificationEventArgs e)

{
    ....
}

//Register event handler for Confirmation-Event
    ....
        tiaPortal.Confirmation += TiaPortal_Confirmation;
....

private static void TiaPortal_Confirmation(object sender, ConfirmationEventArgs e)
{
    ....
}
```

Attributes of TIA Portal notifications

TIA Portal notifications have the following attributes:

Attribute	Description
`Caption`	Returns the name of the confirmation.
`DetailText`	Returns the detail text of the confirmation.
`Icon`	Returns the icon of the confirmation.
`IsHandled`	Returns the confirmation or specifies if it is still pending.
`Text`	Returns the text of the confirmation.

Attributes of confirmations

Confirmations have the following attributes:

Attribute	Description
`Caption`	Returns the name of the confirmation.
`Choices`	Returns the option to acknowledge the confirmation.
`DetailText`	Returns the detail text of the confirmation.
`Icon`	Returns the icon of the confirmation.
`IsHandled`	Returns the confirmation or specifies if it is still pending.
`Result`	Returns the result of the acknowledgment or specifies it.
`Text`	Returns the text of the confirmation.

See also

Program-controlled acknowledgement of dialogs with system events (Page 94)

5.2.11 Program-controlled acknowledgement of dialogs with system events

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- Event handlers are registered.
  See Connecting to the TIA Portal (Page 82)

Application

When you operate the TIA Portal with the user interface, dialogs with system events are displayed for some program sequences. You decide how you want to proceed based on these system events.

When the TIA Portal is accessed with a TIA Portal Openness application, these system events must be acknowledged by means of corresponding ".NET" events.

The permitted confirmations are contained in the `Choices` list:

- `Abort`
- `Cancel`
- `Ignore`
- `No`
- `NoToAll`
- `None`
- `OK`
- `Retry`
- `Yes`
- `YesToAll`

The value of `ConfirmationEventArgs.Result` must be one of the above-mentioned entries. Otherwise, an exception is thrown.

Program code

Modify the following program code to notify the project engineer about executed actions of a TIA Portal Openness application:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;

using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
Namespace
ConfirmDialogsWithSystemMessagesProgrammatically
{
class internal program()
{
private static void Main(string[] args)
{
// To respond to a confirmation event
using (TiaPortal tiaPortal = new TiaPortal());
{
tiaPortal.Confirmation += TiaPortalConfirmation(object sender, ConfirmationEventArgs e);
tiaPortal.Notification += Notification;
try
{
//perform actions that will result in a notification event
}
finally
{
tiaPortal.Notification -= Notification;
}
}
private void TiaPortalConfirmation(object sender, ConfirmationEventArgs e)
{
    ...
}
//Handles notifications
using (TiaPortal tiaPortal = new TiaPortal())
{
tiaPortal.Notification += Notification;
try
{
//perform actions that will result in a notification event
```

```
}
finally
{
tiaPortal.Notification -= Notification;
}
}
}
}
```

5.2.12 Terminating the connection to the TIA Portal

Introduction

If you started the TIA Portal instance without a user interface and if your application is the only TIA Portal Openness client attached to the TIA Portal, you can close the TIA Portal instance with the TIA Portal Openness application. Otherwise, you disconnect the TIA Portal Openness application from the TIA Portal instance.

Use the `IDisposable.Dispose()` method to separate or close the active instance of the TIA Portal.

You can use the `IDisposable.Dispose()` method as follows:

- With a using statement.
- Surround the object description with a try-finally block and call the `IDispose.Dispose()` method within the finally block.

You can no longer access the TIA Portal when you close the active instance of the TIA Portal.

Note

When a configuration engineer closes the TIA Portal instance despite ongoing access of a TIA Portal Openness application, an exception of the class "NonRecoverableException" is thrown in the TIA Portal Openness application on the next API access. You can subscribe to the dispose event to get a call when the TIA Portal is closed.

Program code

Modify the following program code to separate or close the connection to the TIA Portal:

```
// Add code to dispose the application if the application is still instantiated
if (tiaPortal != null)
{
    tiaPortal.Dispose();
}
```

See also

Event handlers (Page 93)

5.2.13 Diagnostic interfaces on TIA Portal

Application

You can retrieve certain diagnostic information from running instances of TIA Portal via a static method. The diagnostic interface is implemented on the TiaPortalProcess object, which can be retrieved for any currently running instance of the TIA Portal.

The diagnostic interface is not blocking, so you can retrieve the TiaPortalProcess object and access it's members, regardless of if the TIA Portal is busy or not. The diagnostic interface includes the following Members:

Class TiaPortalProcess

Member	Type	Function
AcquisitionTime	DateTime	The time when the TiaPortalProcess object was acquired. Since the TiaPortalProcess object represents a completely static snapshot of the state of the TIA Portal at a given point in time, the information it contains may become outdated.
Attach	TiaPortal	Attaches to the given TiaPortalProcess, it returns a TiaPortal instance.
AttachedSessions	IList<TiaPortalSession>	A collection of all other sessions currently attached to the same TIA Portal. This collection can be empty. Each session is represented by a TiaPortalSession object.
Attaching	EventHandler<AttachingEventArgs>	This event enables an application to approve any attempts to attach to the TIA Portal. When another application attempts to attach to the TIA Portal, the subscribers of this event are notified and given 10 seconds to approve the attachment. If any subscriber ignores this event or does not respond in time, it is understood to be denying the other application permission to attach. Crashed applications, being unable to respond to this event and cannot cause an application to be denied permission to attach.

Member	Type	Function
Dispose	void	Closes the associated TIA Portal instance.
Id	int	The Process ID of the TIA Portal.
InstalledSoftware	IList<TiaPortalProduct>	A collection of all the products currently installed as part of the TIA Portal. Each product is represented by a TiaPortalProduct object, which is described below.
Mode	TiaPortalMode	The mode in which the TIA Portal was started. The current values are WithUserInterface and WithoutUserInterface.
Path	FileInfo	The path to the executable of the TIA Portal.
ProjectPath	FileInfo	The path to the project which is currently open in the TIA Portal. If no project is open, this attribute will be null.

Class TiaPortalSession

Member	Type	Function
AccessLevel	TiaPortalAccessLevel	The level access of the session. It is represented as a flags enum, where multiple levels of access are possible. TiaPortalAccessLevel is described in detail below.
AttachTime	DateTime	The time when the connection to the TIA Portal was established.
Id	int	The Id of the current session.
IsActive	bool	Returns "true" if the TIA Portal is currently processing a call from the running session.
Dispose	void	Severs the process's connection to the TIA Portal. This method does not kill the process itself in the way that System.Diagnostics.Process.Kill would do. The application whose connection is terminated will still get a disposed event, but there is no other indication of why the connection was terminated.
ProcessId	int	The process ID of the attached process.
ProcessPath	FileInfo	The path to the executable of the attached process.

Member	Type	Function
TrustAuthority	TiaPortalTrustAuthority	Indicates if the current session was started by a process that was signed, and if it is a TIA Portal Openness certificate or not. TrustAuthority is a flags enum and is described below.
UtilizationTime	TimeSpan	The period of time the process has spent actively using the TIA Portal.Combined with the AttachTime attribute, this could be used to determine usage percentages or similar data.
Version	string	The version of the Siemens.Engineering.dll to which the session is attached to.

Enum TiaPortalAccessLevel

Enum Value	Function
None	This is not a valid value. It is included because TiaPortalAccessLevel is a flags enum which needs an appropriate "zero value" to represent no flags being set, but it will never appear in actual use because no session can be started that has no access.
Published	The session has access to published functionality.
Modify	The session has modify access.

Enum TiaPortalTrustAuthority

Enum Value	Function
None	The main module of the attached process is not signed with a certificate.
Signed	The main module is signed with a certificate, which is not a TIA Portal Openness certificate.
Certified	The main module is signed with a TIA Portal Openness certificate.
CertifiedWithExpiration	The main module is signed with a TIA Portal Openness certificate that will become invalid at the end of its lifetime.

Class TiaPortalProduct

Member	Type	Function
Name	string	The name of the product (e.g. STEP 7 Professional).
Options	IList<TiaPortalProduct>	A collection of all optional packages that belong to the connected TIA Portal, represented as TiaPortalProduct objects. If an option package itself has option packages, this nesting could continue.
Version	string	The version string of the product.

The following code snippet provides an example of how to use the diagnostic Interface to query information and of how to use them in your application.

```
public void TiaPortalDiagnostics()
{
    IList<TiaPortalProcess> tiaPortalProcesses = TiaPortal.GetProcesses();
    foreach (TiaPortalProcess tiaPortalProcess in tiaPortalProcesses)
    {
        Console.WriteLine("Process ID: {0}", tiaPortalProcess.Id);
        Console.WriteLine("Path: {0}", tiaPortalProcess.Path);
        Console.WriteLine("Project: {0}", tiaPortalProcess.ProjectPath);
        Console.WriteLine("Timestamp: {0}", tiaPortalProcess.AcquisitionTime);
        Console.WriteLine("UI Mode: {0}", tiaPortalProcess.Mode);
        //See method body below.
        Console.WriteLine("Installed Software:");
        EnumerateInstalledProducts(tiaPortalProcess.InstalledSoftware);
        Console.WriteLine("Attached Openness Applications:");
        foreach (TiaPortalSession session in tiaPortalProcess.AttachedSessions)
        {
            Console.WriteLine("Process: {0}", session.ProcessPath);
            Console.WriteLine("Process ID: {0}", session.ProcessId);
            DateTime attachTime = session.AttachTime;
            TimeSpan timeSpentAttached = DateTime.Now - attachTime;
            TimeSpan utilizationTime = session.UtilizationTime;
            long percentageTimeUsed = (utilizationTime.Ticks / timeSpentAttached.Ticks) *
100;
            Console.WriteLine("AttachTime: {0}", attachTime);
            Console.WriteLine("Utilization Time: {0}", utilizationTime);
            Console.WriteLine("Time spent attached: {0}", timeSpentAttached);
            Console.WriteLine("Percentage of attached time spent using TIA Portal: {0}",
percentageTimeUsed);
            Console.WriteLine("AccessLevel: {0}", session.AccessLevel);
            Console.WriteLine("TrustAuthority: {0}", session.TrustAuthority);
            if ((session.TrustAuthority & TiaPortalTrustAuthority.Certified) !=
TiaPortalTrustAuthority.Certified)
            {
                Console.WriteLine("TrustAuthority doesn't match required level, attempting
to terminate connection to TIA Portal.");
                session.Dispose();
            }
        }
    }
}
public void EnumerateInstalledProducts(IEnumerable<TiaPortalProduct> products)
{
    foreach (TiaPortalProduct product in products)
    {
        Console.WriteLine("Name: {0}", product.Name);
        Console.WriteLine("Version: {0}", product.Version);
        //recursively enumerate all option packages
        Console.WriteLine("Option Packages \n:");
        EnumerateInstalledProducts(product.Options);
    }
}
```

Security Relevant Information

Because of the fact that no connection to the TIA Portal is needed to use the diagnostics interface, it's possible to write a Windows service that uses the attaching event to check any application attempting to attach to a TIA Portal, e.g. only applications that begin with your company's name are allowed to attach. Another option might be to always grant access, but write information about attaching processes to a log. The following program code is an example event handler to check incoming connections:

```
public void OnAttaching(object sender, AttachingEventArgs e)
{
    string name = Path.GetFileNameWithoutExtension(e.ProcessPath);
    TiaPortalAccessLevel requestedAccessLevel = e.AccessLevel &
TiaPortalAccessLevel.Published;
    TiaPortalTrustAuthority certificateStatus = e.TrustAuthority
&TiaPortalTrustAuthority.Certified;
    if (requestedAccessLevel == TiaPortalAccessLevel.Published &&
       certificateStatus == TiaPortalTrustAuthority.Certified &&
       name.StartsWith("SampleCustomerName"))
    {
        e.GrantAccess();
    }
}
```

5.2.14 Exclusive access

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The "TIA Portal" class provides the method "ExclusiveAccess(String text)" to establish an exclusive access to an attached TIA Portal process. The usage of an exclusive access is highly recommended even if it is not mandatory.

Use "ExclusiveAccess" in an "using" statement to ensure it is disposed attribute even when exceptions occur or the application is shutdown.

Note

Any attempt to create a second exclusive access within the scope of an open exclusive access will result in an recoverable exception being raised.

Modify the following example to get "ExclusiveAccess" to an instance:

```
...
[assembly: AssemblyTitle("MyApplication")]
// This will be used for the exclusive access dialog when present....
TiaPortal tiaPortal = ...;
using (ExclusiveAccess exclusiveAccess = tiaPortal.ExclusiveAccess("My Activity"))
{
    ...
}
```

After acquiring an "ExclusiveAccess" instance for a given TIA Portal process a dialog will be displayed. This dialog will display the message provided during instantiation. In addition the following information of the client application will be displayed:

- the assembly title of the manifest data if available; otherwise, the process name
- the process ID
- the SID (Openness Session ID of the client application's TiaPortal instance)

Note

There can be multiple sessions active for a given TIA Portal Openness client application because there can be multiple instances of TiaPortal each associated with the same TIA Portal process.

The client application can also update the displayed content of the exclusive access dialog by setting the "Text" attribute with new values. Modify the following program code to invoke this behavior:

```
exclusiveAccess = ...;
...
exclusiveAccess.Text = "My Activity Phase 1";
...
exclusiveAccess.Text = "My Activity Phase 2";
...
exclusiveAccess.Text = String.Empty; // or null;
...
```

You can request that the exclusive access could be cancelled by selecting the "Cancel" button. Modify the following program code to invoke this behavior:

```
exclusiveAccess = ...;
...
if (exclusiveAccess.IsCancellationRequested)
{
    // stop your activity
    ...
}
else
{
    // continue your activity
    ...
}
...
```

5.2.15 Dynamic behaviours

Dynamic behaviors in Openness

The Openness users can determine objects and their attributes during the run time. They also are able to invoke actions and navigate to related objects bidirectionally in the hierarchy. In order to support dynamic behaviors the following interfaces are provided and implemented by Openness objects:

IEngineeringInstance

The interface that allows up navigation in the Object Model.

Properties:

- IEngineeringObject Parent - Parent object of an instance.

IEngineeringCompositionOrObject

The interface is implemented by both engineering objects and compositions.

IEngineeringObject

The interface is implemented by the majority of objects that are available for the Openness users. IEngineeringObject interface is derived from IEngineeringCompositionOrObject and IEngineeringInstance:

Methods:

- IList<EngineeringCreationInfo> GetCreationInfos(string compositionName) - Gets the list of composition infos available for the object.
- IEngineeringObject Create(string compositionName, Type type, IEnumerable<KeyValuePair<string, object>> parameters) - Creates an IEngineeringObject of indicated type initialized with values as indicated compositionName within the parameters
- object GetAttribute(string name) - Gets an attribute with the given name.
- IList<EngineeringAttributeInfo> GetAttributeInfos() - Returns a collection of EngineeringAttributeInfo objects describing the different attributes on this object.
- IList<object> GetAttributes(IEnumerable<string> names) - Gets a list of attributes for the given names
- IEngineeringCompositionOrObject GetComposition(string name) - Gets an IEngineeringCompositionOrObject with the given name.
- IList<EngineeringInvocationInfo> GetInvocationInfos() - Returns a collection of EngineeringInvocationInfo objects describing the different actions on this object.
- object Invoke(string name, IEnumerable<KeyValuePair<Type, object>> parameters) - Invokes the method represented by the current instance, using the specified parameters.
- void SetAttribute(string name, object value) - Sets an attribute with the given name to the given value
- void SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes) - Sets the attributes with the given names to the given values as indicated in attributes.
- EngineeringObjectHandle GetHandle() - Gets the object's unique handle.
- void SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes, AttributeDelegate errorHandler) - Sets the attributes with the given names to the given values as indicated in attribute key value pair. If any error occurs (such as, attribute name/ value is invalid), a callback call will be triggered so that you can decide whether to ignore and continue with setting value to the next attribute or abort setting the values to the remaining attributes.

Note

With TIA Portal Openness V18, the new method void SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes, AttributeDelegate errorHandler) is addded to the IEngineeringObject.

Currently the SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes, AttributeDelegate errorHandler) is supported by the following Engineering objects:

- Siemens.Engineering.HW.Features.NetworkInterface
- Siemens.Engineering.HW.Features.NetworkPort
- Siemens.Engineering.HW.Features.PlcAccessLevelProvider
- Siemens.Engineering.HW.Address
- Siemens.Engineering.HW.Channel

- Siemens.Engineering.HW.Device
- Siemens.Engineering.HW.DeviceImpl
- Siemens.Engineering.HW.DeviceItem
- Siemens.Engineering.HW.DeviceItemImpl
- Siemens.Engineering.HW.HardwareObject
- Siemens.Engineering.HW.IoConnector
- Siemens.Engineering.HW.IoController
- Siemens.Engineering.HW.IoSystem
- Siemens.Engineering.HW.MrpDomain
- Siemens.Engineering.HW.Node
- Siemens.Engineering.HW.Subnet
- Siemens.Engineering.HW.SyncDomain
- Siemens.Engineering.HW.TransferArea

If any other Engineering object is used, the method behaves similar to the method void SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes) which has no callback.

Program code: SetAttributes(IEnumerable<KeyValuePair<string, object>>) method

```
...
...
List<KeyValuePair<string, object>> attributeKeyValuePair = new List<KeyValuePair<string,
object>>()
{
new KeyValuePair<string, object>("Name", "DeviceItemName")
};
deviceItem.SetAttributes(attributeKeyValuePair, (AttributeConfiguration
attributeConfiguration) =>
 {
//This callback delegate will be invoked when there is an error while executing the
SetAttributes API,
//reason could be either due to the attribute having invalid attribute name, invalid value,
//invalid type of the value or when tried to set the value to a readonly attribute
switch (attributeConfiguration)
{
case AttributeReadOnly attributeReadOnly:
attributeReadOnly.CurrentSelection = AttributeChoiceSelection.Ignore;
break;
case AttributeNameUnsupported attributeNameUnsupported:
//Attribute name is not supported
attributeNameUnsupported.CurrentSelection = AttributeChoiceSelection.Ignore;
// Or AttributeChoiceSelection.Abort
break;
case AttributeTypeUnsupported attributeTypeUnsupported:
//The the type of supplied value is not supported
attributeTypeUnsupported.CurrentSelection = AttributeChoiceSelection.Ignore; // Or
AttributeChoiceSelection.Abort
break;
case AttributeValueUnsupported attributeValueUnsupported:
//The value is not supported
attributeValueUnsupported.CurrentSelection = AttributeChoiceSelection.Ignore; // Or
AttributeChoiceSelection.Abort
break;
}
}
);
```

IEngineeringRoot

This interface is implemented by TiaPortal objects and indicates that the user is at the top object in the hierarchy. IEngineeringRoot is derived from IEngineeringObject, IEngineeringCompositionOrObject and IEngineeringInstance.

Methods:

- IEngineeringObject GetObject(EngineeringObjectHandle objectHandle) - Gets the object from a handle

IEngineeringComposition

The interface is implemented by the majority of objects that are available for the Openness users. IEngineeringObject interface is derived from IEngineeringCompositionOrObject and IEngineeringInstance. In addition to the properties and methods described in Working with compositions (Page 111), the following methods are supported.

Methods:

- IEngineeringObject Create(Type type, IEnumerable<KeyValuePair<string, object>> parameters) - Creates an IEngineeringObject of indicated type initialized with values as indicated in parameters..
- IList<EngineeringCreationInfo> GetCreationInfos() - Gets the collection of EngineeringCreateInfo objects describing the different CreateInfos on this object.
- IList<EngineeringInvocationInfo> GetInvocationInfos() - Returns a collection of EngineeringInvocationInfo objects describing the different actions on this object.
- object Invoke(string name, IEnumerable<KeyValuePair<Type, object>> parameters) - Invokes the method represented by the current instance, using the specified parameters.

IEngineeringObjectAssociation

This interface is implemented by the collection of associated engineering objects. This interface is derived from IEngineeringAssociation and IEngineeringInstance. See Working with associations (Page 111)

IEngineeringServiceProvider

The interface is implemented by engineering objects that can provide service(s).

Methods:

- T GetService<T>() where T : class, IEngineeringService - Gets an instance of type T.
- IList<EngineeringServiceInfo> GetServiceInfos() - Returns a collection of EngineeringServiceInfo objects describing the different services on this object.

IEngineeringService

The interface representing an engineering service that is provided by IEngineeringServiceProvider.

EngineeringObjectHandle

The structure that provides a unique object handle.

See also

Working with compositions (Page 111)

Working with associations (Page 111)

5.2.16 Working with associations

Accessing associations

An association describes the relationship between two or more objects at type level.

TIA Portal Openness supports access to associations via index and via "foreach" loops. Direct access, for example via string name, is not supported.

Attributes

The following attributes are available:

- `int Count`
- `bool IsReadonly`
- `IEngineeringObject Parent`
- `retType this [ int index ] { get; }`

Methods

TIA Portal Openness supports the following methods:

- `int IndexOf ( type )`: Returns the index in the association for a transferred instance.
- `bool Contains ( type )`: Determines whether the transferred instance is contained in the association.
- `IEnumerator GetEnumerator <retType>()`: Employed within "foreach" loops to access an object.
- `void Add ( type )`[1]: Adds the transferred instance to the association.
- `void Remove ( type )`[1]: Removes the transferred instance from the association.

[1]: Not supported by all associations.

5.2.17 Working with compositions

Accessing compositions

A composition is the special case of an association. A composition expresses a semantic relationship of two objects, of which one is part of the other.

Attributes

The following attributes are available:

- `int Count`
- `bool IsReadonly`

- `IEngineeringObject Parent`
- `retType this [int index] {get;}`: Indexed access to an object of the composition. This type of access should only be used in a targeted manner, as each indexed access operation exceeds process boundaries.

Methods

TIA Portal Openness supports the following methods:

- `retType Create (id, ...)`: Creates a new instance and adds this instance to the composition.
  The signature of the method depends on the way in which the instance is created. This method is not supported by all compositions.
- `type Find (id, ...)`: Scans a composition for the instance with the transferred ID.
  The search is not recursive. The signature of the method depends on the way in which the instance is searched for. This method is not supported by all compositions.
- `IEnumerator GetEnumerator<retType> ()`: Employed within "foreach" loops to access an object.
- `Delete (type)`[1]: Deletes the instance specified by the current object reference.
- `int IndexOf (type)`: Returns the index in the composition for a transferred instance.
- `bool Contains (type)`: Determines whether the transferred instance is contained in the composition.
- `void Import(string path, ImportOptions importOptions)`[1]: Used for each composition that contains importable types.
  Each import signature includes a configuration parameter of the type "ImportOptions (Page 1212)" ("None", "Overwrite") by which the user controls the import behavior.

[1]: Not supported by all compositions.

5.2.18 Verifying object equality

Application

As user of a TIA Portal Openness API, you can check that objects are the same with program code:

- You check whether two object references are the same with the operator "==".
- Use the `System.Object.Equals()` method to check if both objects are really identical with regard to the TIA Portal.

Program code

Modify the following program code to check for object reference types:

```
...
//Composition
DeviceComposition sameCompA = project.Devices;
DeviceComposition sameCompB = project.Devices;
if (sameCompA.Equals(sameCompB))
{
    Console.WriteLine("sameCompA is equal to sameCompB");
}
if (!(sameCompA == sameCompB))
{
    Console.WriteLine("sameCompA is not reference equal to sameCompB");
}
DeviceComposition sameCompAsA = sameCompA;
if (sameCompAsA.Equals(sameCompA))
{
    Console.WriteLine("sameCompAsA is equal to sameCompA");
}
if (sameCompAsA == sameCompA)
{
    Console.WriteLine("sameCompAsA is reference equal to sameCompA");
}
MultiLingualGraphicComposition notSameComp = project.Graphics;
if (!sameCompA.Equals(notSameComp))
{
    Console.WriteLine("sameCompA is not equal to notSameComp");
}
```

5.2.19 Read operations for attributes

Group operations and standard read operations for attributes

TIA Portal Openness supports access to attributes via the following methods which are available at the object level:

- Group operation for read access
- Standard read operations

Program code for group operations

```
//Exercise GetAttributes and GetAttributeNames
//get all available attributes for a device,
//then get the names for those attributes, then display the results.
private static void DynamicTest(Project project)
{
    Device device = project.Devices[0];
    IList<string> attributeNames = new List<string>();
    IList<EngineeringAttributeInfo> attributes =
((IEngineeringObject)device).GetAttributeInfos();
    foreach (EngineeringAttributeInfo engineeringAttributeInfo in attributes)
    {
        string name = engineeringAttributeInfo.Name;
        attributeNames.Add(name);
    }
    IList<object> values = ((IEngineeringObject)device).GetAttributes(attributeNames);
    for (int i = 0; i < attributes.Count; i++)
    {
        Console.WriteLine("attribute name: " + attributeNames[i] + " value: " + values[i]);
    }
}
```

Group operation for read access

This method is available for any object:

```
public abstract IList<object> GetAttributes(IEnumerable<string>
names);
```

Standard read operations

The following operations are available:

- Retrieve the names of available attributes:
  Use the method GetAttributeInfos() (Page 123) on an `IEngineeringObject`.
- Generic method for reading an attribute
  `public abstract object GetAttribute(string name);`

Note

Dynamic attributes are not shown in IntelliSense because their availability depends on the status of the object instance.

5.2.20 Transaction handling

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Operation

A persistence (project, library, etc.) opened within an associated TIA Portal process can be modified by a TIA Portal Openness client application. You can produce this modification from a single operation or by a series of operations. Depending on the activity, it is reasonable to group these operations into a single undo unit for more logical workflows. Additionally there are performance advantages provided by grouping operations into a single undo unit. To support this, the "ExclusiveAccess" class provides the method "Transaction(ITransactionSupport persistence, string undoDescription)". The invocation of this method results in the instantiation of a new disposable object of type "Transaction". You have to provide a description of the transaction's contents (the text attribute cannot be null or Empty). While this instance has not been disposed, all client application operations will be grouped into a in a single undo unit within the associated TIA Portal process.

Modify the following program code to acquire a "Transaction" instance:

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
    ...
}
```

Note

Use a "using" statement to instatiate a "Transaction" to ensure it is disposed properly even when exceptions occur, thus rolling back the transaction.

Consistent commit or rollback

The use of a "Transaction" within a client application helps you to ensure that there is a predictable way to commit or rollback a set of modifications. Your client application must decide whether or not to commit their modifications to a persistence. To do this your application must request that the modifications within the scope of an open transaction be committed when the transaction is disposed by invoking the 'Transaction.CommitOnDispose()' method. If this method is never invoked in the code flow, the modifications within the scope of the open transaction will automatically be rolled back when it is disposed.

Note

Any call to Transaction.CommitOnDispose() made after a Siemens.Engineering.TargetInvocationException has occurred within an open transaction will result in a recoverable exception being thrown.

If an exception occur after making the request, all modifications within the scope of an open transaction will still be rolled back on its disposal.

There are two properties available to determine the committal state of a given transaction:

Property	Data type	Description
Transaction.CanCommit	boolean	Indicates if a prior call to Transaction.CommitOnDispose() will be honored, or if a subsequent call to Transaction.CommitOn-Dispose() will result in a recoverable exception
Transacation.CommitRequested	boolean	Indicates if a call to Transaction.CommitOnDispose() has been made by the user within the given transaction

Modify the following program code to create a single undo unit in the attached TIA Portal containing two "Create" modifications:

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
// Create a single undo unit with name "My Operation" in attached TIA Portal
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation")
{
project.DeviceGroups.Create("My Group 1");
project.DeviceGroups.Create("My Group 2");
transaction.CommitOnDispose();
}
```

Modify the following program code to create no undo unit:

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
project.DeviceGroups.Create("My Folder 1");
project.DeviceGroups.Create("My Folder 2");
}
```

Modify the following program code to create no undo unit in the attached TIA Portal. This is due to exception raised by the second attempt to create a folder "My Folder 1" even though the request to commit on disposal was made before the exception was raised:

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
project.DeviceGroups.Create("My Folder 1");
project.DeviceGroups.Create("My Folder 2");
transaction.CommitOnDispose();
project.DeviceGroups.Create("My Folder 1");
}
```

Modify the following program code to create no undo unit in the attacted TIA Portal. This is due to excpetion raised by the second attempt to create a folder "My Folder 1" even though the try/catch block surrounds the call that caused the exception to be raised:

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
project.DeviceGroups.Create("My Folder 1");
project.DeviceGroups.Create("My Folder 2");
transaction.CommitOnDispose();
try
{
project.DeviceGroups.Create("My Folder 1");
}
catch (Siemens.Engineering.TargetInvocationException ex)
{
...
}
}
```

Modify the following program code to create undo unit created in the attached TIA Portal. This is due to the invocation of Transaction.CommitOnDispose() after a

Siemens.Engineering.TargetInvocationException is raised prior in the code stream even though a try/catch block surrounds the call that caused the exception to be raised:

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
try
{
project.DeviceGroups.Create("My Folder 1");
project.DeviceGroups.Create("My Folder 1");
}
catch (Siemens.Engineering.TargetInvocationException ex)
{
...
}
transaction.CommitOnDispose();
}
```

Modify the following program code to have a transaction rolled back . This is due to the Transaction.CommitOnDispose() never being executed if the execution of any prior statement raises an exception within the scope of the given transaction.

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
try
{
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
project.DeviceGroups.Create("My Folder 1");
project.DeviceGroups.Create("My Folder 1");
transaction.CommitOnDispose();
}
}
catch (Exception ex)
{
...
}
```

Restrictions

The following actions are not allowed inside of a transaction. Calling these will result in a recoverable exception:

- Compile
- Go Online
- Go Offline
- ProjectText Import
- ProjectText Export
- Open Global Library

- Close Global Library
- Project Create
- Project Open
- Project OpenWithUpgrade
- Project Save
- Project SaveAs
- Project Close

Undo behavior

Actions performed by a TIA Portal Openness client application can result in undo units within the attached TIA Portal process. Each of these undo entries will be grouped under a location entry. This location entry will compose the following information from the client application:

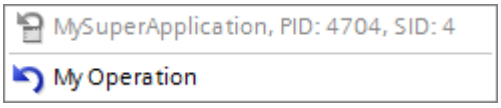
- the assembly title from the manifest data if available; otherwise, the process name
- the process ID
- the SID (Openness Session ID of the client application's TiaPortal instance)
- optionally an indication that the client process is still running

These entries will be one of the following two kinds:

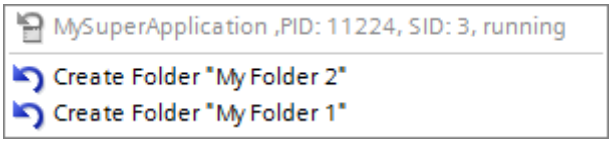
1. The operations that are gathered into one undo transaction as a result of using a "Transaction" have the description as provided by the client application when the "Transaction" was instantiated.
   - Undo entry for a running client application:

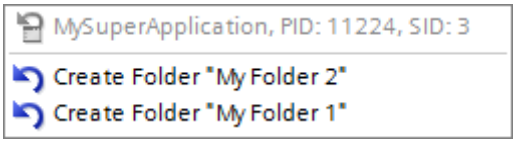
   - Undo entry for a stopped client application:

2. The operations that are executed individually have individual undo entries describing the operation as defined in the respective command meta data.
   - Undo entry for a running client application:

   - Undo entry for a stopped client application:

5.2.21 Creating a DirectoryInfo/FileInfo object

Application

The instances of `DirectoryInfo` and `FileInfo` classes have to contain an absolute path. Otherwise the methods using the `DirectoryInfo` or `FileInfo` objects.will lead to an exception.

Program code

Modify the following program code to create a `DirectoryInfo` or a `FileInfo` object.

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;


using Siemens.Engineering.SW.ExternalSources;

using Siemens.Engineering.SW.Tags;

using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
namespace CreateObjectDirectoryInfoFileInfo
{
internal class Program
{
private static void Main(string[] args)
{
RunTiaPortal();
//..
//Create a DirectoryInfo object
string directoryPath = @"D:\Test\Project 1";
DirectoryInfo directoryInfo = new DirectoryInfo(directoryPath);
//Create a FileInfo object
//please adapt the path and the extension apx to the installed version of TIA Portal
string fileName = @"D:\Test\Project 1\Project 1.apx";
FileInfo fileInfo = new FileInfo(fileName);
//...
}
private static void RunTiaPortal()
{
Console.WriteLine("Starting TIA Portal");
using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
{
Console.WriteLine("TIA Portal has started");
ProjectComposition projects = tiaPortal.Projects;
Console.WriteLine("Opening Project...");
// please adapt the path and the extension apx to the installed version of TIA Portal
FileInfo projectPath = new FileInfo(@"C:\Demo\AnyCompanyProject.apx");
//edit the path according to your project
Project project = null;
```

```
try
{
project = projects.OpenWithUpgrade(projectPath);
}
catch (Exception)
{
Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
Console.WriteLine("Demo complete hit enter to exit");
Console.ReadLine();
return;
}
Console.WriteLine(String.Format("Project {0} is open", project.Path.FullName));
IterateThroughDevices(project);
project.Close();
Console.WriteLine("Demo complete hit enter to exit");
Console.ReadLine();
}
}
private static void IterateThroughDevices(Project project)
{
if (project == null)
{
Console.WriteLine("Project cannot be null");
return;
}
Console.WriteLine(String.Format("Iterate through {0} device(s)", project.Devices.Count));
foreach (Device device in project.Devices)

{
Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
}
Console.WriteLine();
}
}
}
```

5.2.22 Self-description support for attributes, navigators, actions, and services

Application

In TIA Portal Openness each IEngineeringServiceProvider of the TIA Portal Openness API describes its capabilities to potential calls.

Self-description Support on IEngineeringObject

Method Name	Return values
GetCompositionInfos	Returns a collection of EngineeringCompositionInfo objects describing the different compositions of these objects. EngineeringCompositionInfo is described below.
GetAttributeInfos	Returns a collection of EngineeringAttributeInfo objects describing the different attributes of these objects. EngineeringAttributeInfo is described below.
GetInvocationInfos	Returns a collection of EngineeringInvocationInfo objects describing the different actions of these objects. EngineeringInvocationInfo is described below.

Self-description Support on IEngineeringServiceProvider

Method Name	Return values
GetServiceInfos	Returns a collection of EngineeringServiceInfo objects describing the different services of these objects. EngineeringServiceInfo is described below.

Self-description Support on IEngineeringComposition

Method Name	Return values
GetCreationInfos	Returns a collection of EngineeringCreationInfo objects describing the different actions of these objects. EngineeringcreationInfo is described below.

Class EngineeringCompositionInfo

Attribute Name	Return values
Name	The name of the composition

Class EngineeringAttributeInfo

Attribute Name	Return values
AccessMode	The level of access supported by the attribute. This attribute is combinable and is described in detail below.
Name	The name of the attribute.
CreateRelevance	The level to which the attribute is create relevant.
SupportedTypes	The potentially supported types of the attribute.

Class EngineeringInvocationInfo

Attribute Name	Return values
Name	The name of the action.
ParameterInfos	A collection of EngineeringInvocationParameterInfo objects describing any parameters that the action might require. EngineeringInvocationParameterInfo is described below.

Class EngineeringServiceInfo

Attribute Name	Return values
Type	The type of the service as a System.Type object.

Enum AccessMode

Enum Value	Return values
None	This is not a valid option.
Read	The attribute can be read.
Write	The attribute can be written.

Class EngineeringInvocationParameterInfo

Attribute Name	Return values
Name	The name of the parameter.
Type	The type of the parameter as a System.Type object

Class EngineeringCreationInfo

Attribute Name	Return values
Type	The Type property returns a System.Type object
ParameterInfos	The ParameterInfos property returns a collection of EngineeringCreationParameterInfo objects

Class EngineeringCreationParameterInfo

Attribute Name	Return values
Name	The name of the parameter in question

Program code

AccessMode is a flags enum and its values can be combined like the following program code:

```
EngineeringAttributeAccessMode value = EngineeringAttributeAccessMode.Read|
EngineeringAttributeAccessMode.Write;
```

Modify the following program code to find all attributes of an IEngineeringObject and to do changes on the access mode of those attributes.

```
...
IEngineeringObject engineeringObject = ...;
IList<EngineeringAttributeInfo> attributeInfos = engineeringObject.GetAttributeInfos();
foreach(EngineeringAttributeInfo attributeInfo in attributeInfos)
{
    switch (attributeInfo.AccessMode)
        {
        case EngineeringAttributeAccessMode.Read:
            ...
            break;
        case EngineeringAttributeAccessMode.Write:
            ...
            break;
        case EngineeringAttributeAccessMode.Read|EngineeringAttributeAccessMode.Write:
            ...
            break;
        }
}
...
```

Modify the following program code of how Create and GetCreationInfos are intended to be used together.

```
...
Project project = tiaPortal.Projects.Open(projectPath);
IEngineeringComposition deviceGroupComposition = project.DeviceGroups;
EngineeringCreationInfo deviceGroupCreationInfo = deviceGroupComposition.GetCreationInfos()
[0];
EngineeringCreationParameterInfo deviceGroupCreationParameterInfo =
deviceGroupCreationInfo.ParameterInfos[0];
string deviceGroupCreationParameterName = deviceGroupCreationParameterInfo.Name;
string groupName = "Example";
IDictionary<string, object> createParameters = new Dictionary<string, object>
{
 {deviceGroupCreationParameterName, groupName }
};
IEngineeringObject group = deviceGroupComposition.Create(deviceGroupCreationInfo.Type,
createParameters);
...
```

5.2.23 Setting attributes of Blocks, DBs & UDTs

Introduction

It is possible to write and read (general) attributes of the Blocks (written in any programming language), DBs and UDTs via the Openness API.

TechnologicalObjects are internally DBs, so this feature is also valid for TOs.

The same checking rules are to be applied on Name by using SetAttribute of the Openness API on block, DB (TO) or UDT which are applied in the GUI.

In case of rule violation an appropriate user exception will be thrown and the attribute value won't be changed.

This extension of the API doesn't affect the XML export / import.

General Attributes

Legend

- X: The attribute is relevant, write to xml, accessible with API
- -: The attribute is not accessible, the attribute does not exist

Attributes	Datatype	Availability via the API		
		CodeBlock	DB	UDT
AutoNumber	Boolean	X (xml); RW (API)	X (xml); RW (API)	-
CodeModifiedDate	DateTime	X (xml); R (API)	X (xml); R (API)	-
CompileDate	DateTime	X (xml); R (API)	X (xml); R (API)	-
CreationDate	DateTime	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
HeaderAuthor	String	X (xml); R (API)	X (xml); R (API)	-
HeaderFamily	String	X (xml); R (API)	X (xml); R (API)	-
HeaderName	String	X (xml); R (API)	X (xml); R (API)	-
HeaderVersion	System.Version	X (xml); R (API)	X (xml); R (API)	-
InstanceOfName	String	-	X (xml); R (API)	-
Interface	String	X (xml); - (API)	X (xml); - (API)	X (xml); - (API)
InterfaceModifiedDate	DateTime	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
IsConsistent	Boolean	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
IsKnowHowProtected	Boolean	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
MemoryLayout (sofar: Optimization)	MemoryLayout	X (xml); R (API)	X (xml); R (API)	-
ModifiedDate	DateTime	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
Name	String	X (xml); RW (API)	X (xml); RW (API)	X (xml); RW (API)
Number	Int32	X (xml); RW (API)	X (xml); RW (API)	-
ParameterModified	DateTime	X (xml); R (API)	X (xml); R (API)	-
ProgrammingLanguage	ProgrammingLanguage	X (xml); R (API)	X (xml); R (API)	-
SecondaryType	String	X (xml); R (API)	-	-
StructureModified	DateTime	X (xml); R (API)	X (xml); R (API)	-

5.2.24 Notes on performance of TIA Portal Openness

Root objects

You can specify several root objects in import files.

Example: You can create several text lists in an XML file instead of one text list for each XML file.

TIA Portal Openness functions

The first call of a TIA Portal Openness function can take longer than any subsequent calls of the TIA Portal Openness function.

Example: If you perform multiple configuration data exports one after the other, the first export can take longer than the subsequent exports.

5.3 Functions for projects and project data

5.3.1 Opening a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- The project to be opened is not open in any other instance of the TIA Portal.

Note

Undo of project upgrade

If you undo an upgrade of a project to V14SP1 after you have connected it to TIA Portal Openness, conflicts will occur.

Application

Use the `Projects.Open` method to open a project. Enter a path to the desired project in the `Projects.Open` method.

The `Projects.Open` method only accesses projects that were created with the current version of TIA Portal or which have been upgraded to the current version. If you access a project of a previous version with the `Projects.Open` method, an exception will be

returned. Use the `OpenWithUpgrade` method, to open projects which have been made with previous versions of TIA Portal.

Note

No access to read-only projects

TIA Portal Openness can only access projects with read and write privileges.

Program code

Modify the following program code to open a project:

```
Project project =tiaPortal.Projects.Open(new FileInfo(@"D:\Project_1\Project_1.apXX"));
if (project != null)
{
try
{
...
}
finally
{
project.Close();
}
}
```

Opening a UMAC protected project

You can also open a UMAC protected project. The overload of Open function takes an additional parameter of type UmacDelegate. This additional parameter allows the caller to specify a handler to be used during UMAC authentication. The new UmacDelegate is implemented with a method containing one parameter of type UmacCredentials. The UmacCredentials has two properties, 'Name' of type string, and 'Type' of type UmacUserType; and one method SetPassword with one parameter of type SecureString. The use of UmacUserType.Project indicates a UMAC scope of project whereas the use of UmacUserType.Global indicates a UMAC scope of application (i.e. controlled by a UMAC server).

Program code

```
...
Siemens.Engineering.Project project = tiaPortal.Projects.Open(new
FileInfo(@"D:\Project_3\Project_2.apXX"), MyUmacDelegate);
if (project != null)
{
try
{
...
}
finally
{
project.Close();
}
}
...
private static void MyUmacDelegate(UmacCredentials umacCredentials)
{
SecureString password = ...; // Get password from a secure location
umacCredentials.Type = UmacUserType.Project;
umacCredentials.Name = "SomeUser";
umacCredentials.SetPassword(password);
}
...
}
```

Opening a UMAC protected project using new authentication

Before TIA Portal Openness V17, You can open the protected project by passing the user name and password through Umac delegate in Open Project API.

With TIA Portal Openness V17, UMAC have introduced new authentication mechanism by which you will able to open a protected project. The following new authentication mechanisms are Desktop Single Sign-On (SSO) and Anonymous User.

TIA Potal Openness V17 supports the following authentication mechanisms. .The Openness implementation is independent of settings in TIA Portal.

- Desktop Single Sign-On (SSO)
- Anonymous User
- Interactive log on
- Credential (Existing From TIA Portal V15.1)

In order to accommodate the additional authentication methods, Openness have introduced a new event name "Authentication" in TIA Portal object. The changes are introduced as a part of V17 Engineering. The Event "Authentication" when registered by the user in the

Openness script, will be executed when open project is called. The Event "Authentication" will be executed only for Open Protected Project.

```
var tiaPortal = new TiaPortal();
tiaPortal.Authentication += OnAuthentication;
private static void OnAuthentication(object sender, AuthenticationEventArgs e)
{
e.AuthenticationTypeProvider = AuthenticationTypeProvider.Credentials;
}
```

You are be able to use the Additional Authentication Methods using the AuthenticationEventArgs event args. Using AuthenticationTypeProvider ,the mode of authentication is provided. From the above example , the crediantial mode is set.

Below are the values for AuthenticationTypeProvider.

- AuthenticationTypeProvider.DesktopSso : Open project with Single sign on user , no password asked
- AuthenticationTypeProvider.Anonymous : Open project with anonymous user , no password asked
- AuthenticationTypeProvider.Interactive : Open project with Interactive logon, User name and password to be given in user interface
- AuthenticationTypeProvider.Credentials : Open Project with User name and password given.

The event is applicable for opening the project in multiuser user environment.The above authentication methods is applicable when a protected project is opened in local session or server view.

If there are exceptions for example, a single sign-on session or a invalid user name / password scenarios are handled by EngineeringTargetInvocationException.

Opening multiple projects

You can open one primary project and multiple secondary projects at time in an instance of the TIA Portal. You can open a project as a primary or a secondary project. If a project is opened as primary, then this project is represented in the project navigation if the Openness application is attached to a TIA Portal and project is opened as secondary, then this project is not reflected in the user interface. Secondary projects are always opened as read-only. a user with read and write priviliges to UMAC projected project will have read-only priviliges when project is opened as secondary. A primary project does not need to be open in order to open a secondary project.

Any opened projects can be enumerated by using the ProjectComposition available on the TiaPortal instance. The order of the projects in the composition will be determined by the order in which the projects were opened. If a project is closed, the index of all projects will be recalculated.

Program code

```
TiaPortal tiaPortal = ...;
Project project1 = tiaPortal.Projects.Open(new
FileInfo(@"D:\Project_1\Project_1.apXX"), null, ProjectOpenMode.Primary);
Project project3 = tiaPortal.Projects.Open(new
FileInfo(@"D:\Project_3\Project_3.apXX"), null, ProjectOpenMode.Secondary);
bool isPrimary = project3.IsPrimary
```

Opening projects created with previous versions

Use the `OpenWithUpgrade` method to open a project which was created with the previous version of TIA Portal. The method will create a new, upgraded project and open it.

If you access a project created with an elder version than the previous, an exception will be returned.

Note

If you access a project created with the current version the project will just be opened.

Program code

Modify the following program code to open a project via the `OpenWithUpgrade` method:

```
Project project = tiaPortal.Projects.OpenWithUpgrade(new
FileInfo(@"D:\Some\Path\Here\Project.apXX"));
if (project != null)
{
    try
    {
        ...
    }
    finally
    {
        project.Close();
    }
}
```

Program code for UMAC protected project

You can also open a UMAC protected which has been created with a previous version of TIA Portal. An overload function of OpenWithUpgrade takes an additional parameter of type UmacDelegate. OpenWithUpgrade is also supported for secondary projects.

```
    ...
    Siemens.Engineering.Project project = tiaPortal.Projects.OpenWithUpgrade(new
FileInfo(@"D:\Project_1\Project.apXX"), MyUmacDelegate);
    ...
```

5.3.2 Creating a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)

Application

Projects can be created via TIA Portal Openness API

- by calling the Create method on ProjectComposition
- by calling the Create method on IEngineeringComposition

ProjectComposition.Create

Modify the following program code:

```
TiaPortal tiaPortal = ...;
ProjectComposition projectComposition = tiaPortal.Projects;
DirectoryInfo targetDirectory = new DirectoryInfo(@"D:\TiaProjects");
// Create a project with name MyProject
Project project = projectComposition.Create(targetDirectory, "MyProject");
```

According to this example

- a folder "D:\TiaProjects\MyProject" will be created.
- a project file "D:\TiaProjects\MyProject\MyProject.aPXX" will be created.

Note

About parameter targetDirectory

The parameter targetDirectory can also represent an UNC (Universal Naming Convention) path, therefore a project can also be created on a network shared drive.

IEngineeringComposition.Create

Modify the following program code:

```
TiaPortal tiaPortal = ...;
ProjectComposition projectComposition = tiaPortal.Projects;

//allows the user to give optional create parameters like author, comment in addition to
mandatory create parameters (targetdirectory, projectname)

IEnumerable<KeyValuePair<string, object>> createParameters = new [] {
           new KeyValuePair<string, object>("TargetDirectory", new
DirectoryInfo(@"D:\TiaProjects")), // Mandatory
           new KeyValuePair<string, object>("Name", "MyProject"), // Mandatory
           new KeyValuePair<string, object>("Author", "Bob"), // Optional
           new KeyValuePair<string, object>("Comment", "This project was created with
Openness") // Optional };

// Create a project with both mandatory and optional parameters
((IEngineeringComposition)projectComposition).Create(typeof (Project), createParameters);
```

According to this example:

- a folder "D:\TiaProjects\MyProject" will be created.
- a project file "D:\TiaProjects\MyProject\MyProject.aPXX" will be created with project attributes Author as "Bob" and Comment as "This project was created with openness".

Parameters for creating project with optional project attributes

Parameter	Data Type	Is Mandatory	Description
Author	String	No	Author of a project.
Comment	String	No	Comment for the project.
Name	String	Yes	Name of a project,
TargetDirectory	DirectoryInfo	Yes	Directory that will contain the created proeject folder.

5.3.3 Accessing general settings of the TIA Portal

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Via TIA Portal Openness you can access general settings of the TIA Portal:

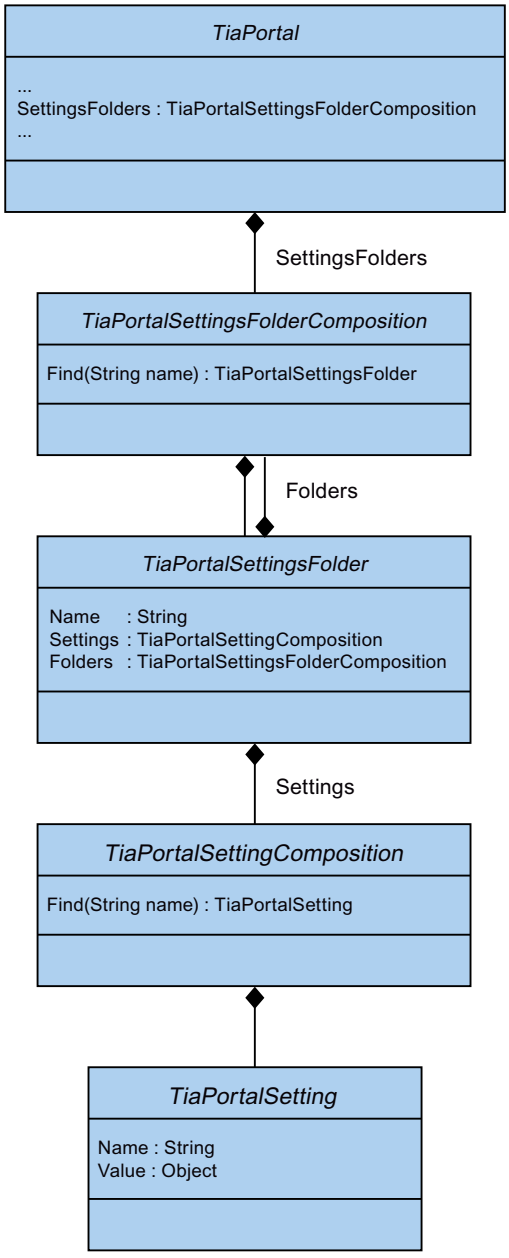
- Current user interface language
- "Search in project" option to create the search index needed for searching within a project.

The following table shows the details of the accessible settings in the "General" section of the TIA Portal settings. The `TiaPortalSettingsFolder` instance will have the name "General".

Setting name	Data type	Writeable	Description
`"SearchInProject"`	System.Boolean	r/w	Enables or disables the creation of the search index needed for searching within a project.
`"UserInterfaceLanguage"`	System.CultureInfo	r/w	Indicates the active user interface language of the TIA Portal or the specification of the active user interface language.

The access to these settings is provided via the `TiaPortalSettingsFolder` class. The `TiaPortalSettingsFolder` class will be accessible via the `Settings` attribute on the `TiaPortal` class.

The following figure shows the specific settings in TIA Portal Openness:


TiaPortal
...
SettingsFolders : TiaPortalSettingsFolderComposition
...
SettingsFolders
TiaPortalSettingsFolderComposition
Find(String name) : TiaPortalSettingsFolder
Folders
TiaPortalSettingsFolder
Name : String
Settings : TiaPortalSettingComposition
Folders : TiaPortalSettingsFolderComposition
Settings
TiaPortalSettingComposition
Find(String name) : TiaPortalSetting
TiaPortalSetting
Name : String
Value : Object

Program code: Search in project

Modify the following program code to activate/deactivate the "Search in project" option.

```
private static void SetSearchInPoject(Project project)
{
    TiaPortalSettingsFolder generalSettingsFolder =
tiaPortal.SettingsFolders.Find("General");
    TiaPortalSetting searchSetting =
generalSettingsFolder.Settings.Find("SearchInProject");

        if (((bool)searchSetting.Value))
        {
            searchSetting.Value = false;
        }
   }
```

Program code: User interface language

Modify the following program code to access the current user interface language.

```
private static void SetUILanguage(Project project)
{

    TiaPortalSettingsFolder generalSettingsFolder =
tiaPortal.SettingsFolders.Find("General");

    TiaPortalSetting UILanguageSetting =
generalSettingsFolder.Settings.Find("UserInterfaceLanguage");

        if (((CultureInfo)UILanguageSetting.Value) != CultureInfo.GetCultureInfo("de-DE"))
        {
            UILanguageSetting .Value = CultureInfo.GetCultureInfo("de-DE");
        }

}
```

See also

Opening a project (Page 128)

5.3.4 Archiving and Retrieving a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)
- A project is save
  See Saving a Project (Page 157)

Application

You can use the TIA Portal Openness to archive the opened and saved project state prior to further any changes and then later you can retrieve the archived project.

Archiving a project

You can archive a TIA Portal project using TIA Openness API available on Siemens.Engineering.Project object.

```
public void Archive(System.IO.DirectoryInfo targetDirectory, string targetName,
Siemens.Engineering.ProjectArchivationMode archivationMode)
```

'targetName' is the name of the file created may it be archived or non archived. This file may or may not contain any file extensions. For ProjectArchivationMode value as Compressed and DiscardRestorableDataAndCompressed, the archived file name is as it is provided by you. If you don't provide any extension or extension apart from "zap15_1" or "zap14" etc., then this archived project could not be retrieved from TIA Portal outside the Openness API. If you don't not choose Compressed or DiscardRestorableDataAndCompressed the result will be the default file structure of a project of the current version.

Note

You must have saved the project before calling Archiving API. In case the project contains any unsaved changes, archive would throw an EngineeringTargetInvocationException.

Project ArchivationMode

The **ProjectArchivationMode** enumeration have four values.

ProjectArchivationMode	Description
None	• No special action are taken with the orginial files. Mode is similiar to a "save as" operation. • No compressed zip file is created in this mode. • The difference with SaveAs in this case is that the Archive will not change the persistence location to the new Archived folder whereas SaveAs does that
DiscardRestorableData	• The file storage stores the project in an internal data file and this file grows whenever a project data modification happens. In the case of DiscardRestorableData mode this data file is reorganized (only latest version of the objects are stored and history is removed from the file) and Intermediate data, the files of the IM directory and tmp directory are not copied to the archive location. • No compressed zip file is created in this mode.
Compressed	• The tmp project folder structure, created by Archiving, is compressed into a zip compatible archive. The tmp folder structure is removed after creation of the zip file.
DiscardRestorableDataAndCompressed	• In the tmp project folder structure, created by Archiving, the restorable data is discarded and then compressed into a zip compatible archive. The tmp folder structure is removed after creation of the zip file.

Program code: Archiving a poject

Modify the following program code to access the archive project without an extension:

```
public void ArchivingProjectWithoutExtension()
{
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
var projectFilePath = @"E:\Sample1\Sample1.ap15_1";
var project = tiaPortal.Projects.Open(new FileInfo(projectFilePath));
var archiveDirectory = @"E:\Archive";
project.Archive(new DirectoryInfo(archiveDirectory), "ArchiveProjectNameWithoutExtension",
ProjectArchivationMode.Compressed);
}
```

Modify the following program code to access the archive project with an extension as '.archive':

```
public void ArchivingProjectWithExtension()
{
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
var projectFilePath = @"E:\Sample1\Sample1.ap15_1";
var project = tiaPortal.Projects.Open(new FileInfo(projectFilePath));
var archiveDirectory = @"E:\Archive";
project.Archive(new DirectoryInfo (archiveDirectory), "ArchiveProjectName.archive",
ProjectArchivationMode.Compressed);
}
```

Note

You are free to use any extension. The behavior of Archive will be same with/without extension.

Retrieving a project

You can use the Openness API to retrieve an archived TIA Portal project. You can only retrieve a compressed archive. For Archived project with 'ProjectArchivationMode.None' or 'ProjectArchivationMode.DiscardRestorableData' enumeration value cannot be retrieved by this API. This API is available on the "Siemens.Engineering.ProjectComposition" object.

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory)
```

For retrieving a UMAC protected project an overloaded API with the UmacDelegate is provided for getting the credentials, then the following API definition is used

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate)
```

Retrieve also supports ProjectOpenMode that can be specified as "Primary" or "Secondary", then the following API definition is used

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory,Siemens.Engineering.UmacDelegate umacDelegate,
Siemens.Engineering.ProjectOpenMode projectOpenMode)
```

Note

umacDelegate could be passed as null if the project is not protected

If the archived project is of a previous TIA Portal version then you have to call the RetrieveWithUpgrade API, then the following API definition is used

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory);
```

If the archived project is of a previous TIA Portal version and UMAC protected then another overloaded API with UmacDelegate is available, then the following API definition is used.

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate)
```

If the archived project is of a previous TIA portal version and project open mode also needs to be specified, then the following API definition is used.

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory,Siemens.Engineering.UmacDelegate umacDelegate,
Siemens.Engineering.ProjectOpenMode projectOpenMode)
```

Note

umacDelegate could be passed as null if the project is not protected. If the project is protected then the user credentials of an admin user is required to perform the RetrieveWithUpgrade action

Program code: Retrieving a project

Modify the following program code to access the retrieve project:

```
public void RetrieveProject()
{
var archivePath = @"E:\Archive\Sample1.zap15_1";
var retrievedProjectsDirectory = @"E:\RetrievedProjects";
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
tiaPortal.Projects.Retrieve(new FileInfo(archivePath), new
DirectoryInfo(retrievedProjectsDirectory));
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Saving a project (Page 157)

5.3.5 Accessing read-only TIA Portal project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

Using TIA Portal Openness, you can perform select operations while working with a read-only TIA Portal project. You can have access to read-only project, but you will not be able to use full set of features that are available to a user with read-write access. For example, a user with read-only credentials can use Openness to open a UMAC protected project as described in Opening a project (Page 128). This functionality does not include Reference projects.

The list of Openness features that are available to you while accessing a read-only project can be categorized into two sets of features - Inherent and Enabled non-modifying actions:

Inherent functionality

- GetAttribute(s) or using the getter for any attribute on any accessible object
- GetComposition on any accessible object
- GetService on any accessible object
- Find actions on any accessible object
- Navigation on any accessible object
- Determining the existence of accessible objects and accessing those objects in compositions and associations
- System.Object methods on any accessible object

Enabled non-modifying actions

- Project.Close (...)
- PlcBlock.ShowInEditor ()
- CaxProvider.Export (Device,...)
- CaxProvider.Export (Project,...)

See also

Opening a project (Page 128)

5.3.6 Accessing languages

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

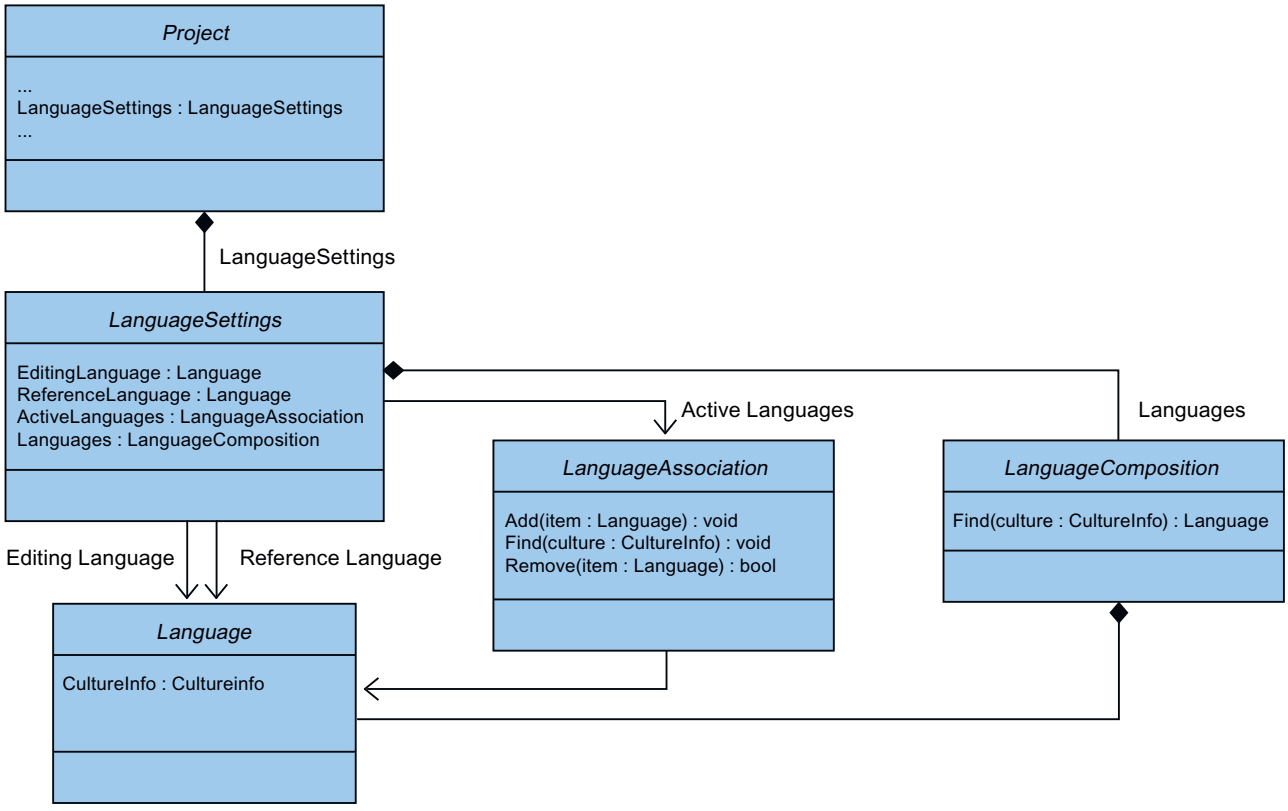
Application

In the TIA Portal you can set and manage the project language in the Editor "Project languages".

TIA Portal Openness supports the following access to the project languages:

- Iterating through supported languages.
- Searching through the collection of supported languages by `System.Globalization.CultureInfo`.
- Accessing individual languages. Each Language object will contain a single read-only attribute `Culture` of type `System.Globalization.CultureInfo`.
- Accessing a collection of active languages.
- Searchring through the collection of active languages by `System.Globalization.CultureInfo`.
- Adding a language to a collection of active languages.
- Removing a language from a collection of active languages.
- Setting an editing language.
- Setting a reference language.

The functionalities are provided by the `LanguageSettings` object. The following figure shows model, which is provided by TIA Portal Openness:



Program code: Setting languages

Modify the following program code to set a language. If you set an inactive language via TIA Portal Openness, the language will be added to the active languages collection.

```
public void SettingLanguages()
{
Project project = ...;
LanguageSettings languageSettings = project.LanguageSettings;
LanguageComposition supportedLanguages = languageSettings.Languages;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
Language supportedGermanLanguage = supportedLanguages.Find(CultureInfo.GetCultureInfo("de-
DE"));
activeLanguages.Add(supportedGermanLanguage);
languageSettings.EditingLanguage = supportedGermanLanguage;
languageSettings.ReferenceLanguage = supportedGermanLanguage;
}
```

Program code: Deactivating an active language

Modify the following program code to deactivate an active language. If you deactivate a language which is used as reference or editing language the language selected will be consistent with the behavior in the user interface.

```
public void DeactivatingLanguages()
{
Project project = ...;
LanguageSettings languageSettings = project.LanguageSettings;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
Language activeGermanLanguage = activeLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
activeLanguages.Remove(activeGermanLanguage);
}
```

See also

Hierarchy of hardware objects of the object model (Page 65)

5.3.7 Determining the object structure and attributes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can determine the navigation structure through the object hierarchy with the `IEngineeringObject` interface. The result is returned as a list:

- Child objects
- Child compositions
- All attributes

Signature

Use the `GetAttributeInfos` method to determine attributes.

```
IList<EngineeringAttributeInfo>
IEngineeringObject.GetAttributeInfos();
```

Program code: Determining objects or compositions

Use the following program code to display alll composition names:

```
public static void DisplayCompositionInfos(IEngineeringObject obj)
{
IList<EngineeringCompositionInfo> compositionInfos = obj.GetCompositionInfos();
foreach (EngineeringCompositionInfo compositionInfo in compositionInfos)
{
Console.WriteLine(compositionInfo.Name);
}
}
```

Modify the following program code if you know the return value:

```
public static DeviceItemComposition GetDeviceItemComposition(Device device)
{
IEngineeringCompositionOrObject composition = ((IEngineeringObject)
device).GetComposition("DeviceItems");
DeviceItemComposition deviceItemComposition = (DeviceItemComposition)composition;
return deviceItemComposition;
}
```

Program code: Determining attributes

Modify the following program code to return attributes of an object with specific access rights in a list:

```
public static void DisplayAttributenInfos(IEngineeringObject obj)
{
IList<EngineeringAttributeInfo> attributeInfos = obj.GetAttributeInfos();
foreach (EngineeringAttributeInfo attributeInfo in attributeInfos)
{
Console.WriteLine("Attribute: {0} - AccessMode {1} ", attributeInfo.Name,
attributeInfo.AccessMode);
switch (attributeInfo.AccessMode)
{
case EngineeringAttributeAccessMode.Read: Console.WriteLine("Attribute: {0} - Read
Access", attributeInfo.Name);
break;
case EngineeringAttributeAccessMode.Write: Console.WriteLine("Attribute: {0} - Write
Access", attributeInfo.Name);
break;
case EngineeringAttributeAccessMode.Read | EngineeringAttributeAccessMode.Write:
Console.WriteLine("Attribute: {0} - Read and Write Access",
attributeInfo.Name);
break;
}
}
}
public static string GetNameAttribute(IEngineeringObject obj)
{
Object nameAttribute = obj.GetAttribute("Name");
return (string)nameAttribute;
}
```

5.3.8 Access software target

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Programm code

Modify the following programm code to make a software target available:

```
public void AccessSoftwareTarget()
{
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<SoftwareContainer>();
if (softwareContainer != null)
{
Software software = softwareContainer.Software;
}
}
```

Modify the following programm code to access the software attributes:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<SoftwareContainer>();
if (softwareContainer != null)
{
    PlcSoftware software = softwareContainer.Software as PlcSoftware;
    string name = software.Name;
}
```

5.3.9 Accessing and enumerating multilingual texts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Multilingual texts in the TIA Portal are e. g. Project.Comment, PlcTag.Comment etc. In TIA Portal Openness, the multilingual texts are represented by the `MultilingualText` object.
A `MultilingualText` object is composed of `MultilingualTextItemComposition`.

`MultilingualTextItemComposition` supports the following `Find` method:

- `Find(<language: Siemens.Engineering.Language>):MultilingualTextItem`

Each `MultilingualTextItem` provides the following attributes:

Attribute name	Data type	Writable	Description
Language	Siemens.Engineering.Language	r/o	Language of this item.
Text	System.String	r/w	Text provided for this language.

Program code: Set multilingual text

```
...
    Language englishLanguage = project.LanguageSettings.Languages.Find(new CultureInfo("en-
US"));
    MultilingualText comment = project.Comment;
    MultilingualTextItemComposition mltItemComposition = comment.Items;
    MultilingualTextItem englishComment = mltItemComposition.Find(englishLanguage);
    englishComment.Text = "English comment";
...
```

Program code: Set multilingual text for devices

Modify the following program code to set the multilingual text for devices and device items:

```
...
    var mltObject = device.GetAttribute("CommentML");
    MultilingualText multilingualText = mltObject as MultilingualText;
    if (multilingualText != null)
    {
        Language englishLanguage = project.LanguageSettings.Languages.Find(new
CultureInfo("en-US"));
        MultilingualTextItem multilingualTextItem =
multilingualText.Items.Find(englishLanguage);
        if (multilingualTextItem != null)
        {
            multilingualTextItem.Text = comment;
        }
    }
...
```

5.3.10 Updating project properties

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to update the "simulation support" property of a project to create a plc program. The setting for simulation support property is available for virtual SINUMERIK controller and SIMATIC PLCs simulated with PLCSIM Advanced .

The following attribute is supported in Openness to access the "simulation support" property of a TIA Portal project. The API is available on the "Siemens.Engineering.Project" object.

Attribute name	Data type	Access	Description
IsSimulationDuringBlockCompilationEnabled	System.Boolean	Read / Write	To indicate whether support for simulation during block compilation is enabled for the project
IsVirtualPlcDuringBlockCompilationEnabled	System.Boolean	Read / Write	To indicate whether support for virtual plc during block compilation is enabled for project

Program code

Modify and use the following program code to access the attribute value:

```
Project project = ...;
// Read the attribute value
var attributeValue = project.IsSimulationDuringBlockCompilationEnabled;
var attributeValue = project.IsVirtualPlcDuringBlockCompilationEnabled;
//Write the attribute value to false
project.IsSimulationDuringBlockCompilationEnabled = false;
//Write the attribute value to True
project.IsVirtualPlcDuringBlockCompilationEnabled = true;
```

See also

Opening a project (Page 128)

5.3.11 Read project related attributes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

By using this function you can get project related attributes from the TIA Portal Openness API. The provided information contains project attributes, project history and products utilized by the project.

Project attributes

The project attributes provide the following information:

Attribute name	Data type	Writeable	Description
Author	System.String	r/o	The author of the project
Comment	Siemens.Engineering.MultilingualText	r/o	The comment of the project
Copyright	System.String	r/o	The copyright statement of the project
CreationTime	System.DateTime	r/o	The time when project has been created
Family	System.String	r/o	The family of the project
IsModified	System.Boolean	r/o	Returns true if the project has been modified
LanguageSettings	Siemens.Engineering.LanguageSettings	r/o	Handles proct languages
LastModified	System.DateTime	r/o	The time when project was last modified
LastModifiedBy	System.String	r/o	Who made the last modification
Name	System.String	r/o	The name of the project
Path	System.IO.FileInfo	r/o	The absolute path of the project
Size	System.Int64	r/o	The size of the project in KB
Version	System.String	r/o	The version of the project

Modify the following program code to access project related attributes:

```
Project project = ...;
string author = project.Author;
string name = project.Name;
string path = project.Path;
DateTime creationTime = project.CreationTime;
DateTime modificationTime = project.LastModified;
string lastModifiedBy = project.LastModifiedBy;
string version = project.Version;
MultilingualText comment = project.Comment;
string copyright = project.Copyright;
string family = project.Family;
Int64 size = project.Size;
LanguageSettings languageSettings = project.LanguageSettings;
```

Modify the following programm code to enumerate the project languages:

```
Project project = ...;
LanguageComposition languages = project.LanguageSettings.Languages;
foreach (Language language in languages)
{
    CultureInfo lang = language.Culture;
}
```

Modify the following program code to get comment text:

```
Project project = ...;
Language english =
project.LanguageSettings.ActiveLanguages.Find(CultureInfo.GetCultureInfo("en-US"));

MultilingualText projectComment = project.Comment;
MultilingualTextItem textItem = project.Comment.Items.Find(english);
string text = textItem.Text;
```

Project history

The project history is a composition of `HistoryEntry` objects, which contain the following information:

Attribute name	Data type	Writeable	Description
Text	System.String	r/o	The event description
DateTime	System.DateTime	r/o	The time when the event was occured

Modify the following program code to enumerate through HistoryEntries and access their attributes:

```
Project project = ...;
HistoryEntryComposition historyEntryComposition = project.HistoryEntries;
foreach (HistoryEntry historyEntry in historyEntryComposition)
{
    string entryText = historyEntry.Text;
    DateTime entryTime = historyEntry.DateTime;
}
```

Note

The text attribute of `HistoryEntry` contains a string in the same language as UI. If a TIA Portal Openness application is attached to a TIA Portal with no UI, the string is always in English

Used Products

The object `UsedProduct` includes the following information:

Attribute name	Data type	Writeable	Description
Name	System.String	r/o	The name of the product used
Version	System.String	r/o	The version of the product

Modify the following program code to enumerate through `UsedProduct` and access the attributes.

```
Project project = ...;
UsedProductComposition usedProductComposition = project.UsedProducts;
foreach (UsedProduct usedProduct in usedProductComposition)
{
    string productName = usedProduct.Name;
    string productVersion = usedProduct.Version;
}
```

5.3.12 Deleting project graphics

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to delete a project graphics:

```
//Deletes a single project graphic entry
public static void DeletesSingleProjectGraphicEntry(Project project)
{
MultiLingualGraphicComposition graphicsAggregation = project.Graphics;
MultiLingualGraphic graphic = graphicsAggregation.Find("Graphic XYZ");
graphic.Delete();
}
```

5.3.13 Compiling a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- All devices are "Offline".

Application

The API interface supports the compilation of devices and program blocks. The compilation result is returned as an object. Depending on type of the object HW or SW or HW/SW compilation will be provided. The following object types are supported:

- `Device – HW & SW`
- `Device` with failsafe CPU - `SW` with switched-off F-activation property
- `DeviceItem – HW`
- `CodeBlock – SW`
- `DataBlock – SW`
- `HmiTarget – SW`
- `PlcSoftware – SW`
- `PlcType – SW`
- `PlcBlockSystemGroup – SW`
- `PlcBlockUserGroup – SW`
- `PlcTypeSystemGroup – SW`
- `PlcTypeUserGroup – SW`

Note

Time stamp format

All time stamps are in UTC. If you want to see the local time you can use DateTime.ToLocalTime().

Note

If a password has been assigned (Page 1193) to the safety program, you are required to a log on to the safety program (Page 1195) to compile software changes. Otherwise, the process is canceled and an exception is triggered.

Signature

Use the `ICompilable` method for compilation.

```
ICompilable compileService =
iEngineeringServiceProvider.GetService<ICompilable>();

CompilerResult result = compileService.Compile();
```

Note

All devices must be "Offline" before you start compiling.

Program code

Modify the following program code to compile the software changes of an object of the type `HmiTarget`:

```
public static void CompileHmiTarget(HmiTarget hmiTarget)
{
ICompilable compileService = hmiTarget.GetService<ICompilable>();
CompilerResult result = compileService.Compile();
}
```

Modify the following program code to compile the software changes of an object of the type `PlcSoftware`:

```
public static void CompilePlcSoftware(PlcSoftware plcSoftware)
{
ICompilable compileService = plcSoftware.GetService<ICompilable>();
CompilerResult result = compileService.Compile();
}
```

Modify the following program code to compile the software changes of an object of the type `CodeBlock`:

```
public static void CompileCodeBlock(PlcSoftware plcSoftware)
{
CodeBlock block = plcSoftware.BlockGroup.Blocks.Find("MyCodeBlock") as CodeBlock;
if (block != null)
{
ICompilable compileService = block.GetService<ICompilable>();
CompilerResult result = compileService.Compile();
}
}
```

Modify the following program code to evaluate the compilation result:

```
private void WriteCompilerResults(CompilerResult result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);
RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(CompilerResultMessageComposition messages, string
indent = "")
{
indent += "\t";
foreach (CompilerResultMessage message in messages)
{
Console.WriteLine(indent + "Path: " + message.Path);
Console.WriteLine(indent + "DateTime: " + message.DateTime);
Console.WriteLine(indent + "State: " + message.State);
Console.WriteLine(indent + "Description: " + message.Description);
Console.WriteLine(indent + "Warning Count: " + message.WarningCount);
Console.WriteLine(indent + "Error Count: " + message.ErrorCount);
RecursivelyWriteMessages(message.Messages, indent);
}
}
```

5.3.14 Modifying project in UMAC

Introduction

You can use the TIA Portal Openness to modify project in user management and access right. With this Function Right , When a project is protected, "Modify Project via Openness API" can be assigned to a role which in turn can be assigned to user.

There are certain behavior changes in TIA Portal Openness when you are having / without having this function right in a protected project and affected when a protect project is opened with a particular user credentials. The changes for Openness API is applicable only on protected project and on project which is opened with ProjectOpenMode as primary.The ProjectOpenMode as secondary is not considered.

Changes in behaviors with function right

The below described behaviors with Openness function right is applicable for all the versions of Siemens.Engineering.dll (V15, V15.1, V16 and V17) if it is running against TIA Portal V17.

- When you are having "Modify project via Openness API" function right along with Project Read Write rights , you are given with full access to Openness API i.e you can perform any operation using Openness API.
- When you are not having "Modify project via Openness API" function right along with Project Read Write rights / Project Read rights, you are entitled to Read only operations in Openness API. The examples of read only operations are export, navigation to a device etc. In this scenario, if you want to do an operation like import which is a data oriented operation (Change of project data in TIA portal) , then you would get an exception depending on the function rights assigned to you.
- When there are user interface related Openness API's example (ShowInEditor), the user interface related function rights does not depend on "Modify project via Openness API" , it depends on GUI function right which should be handled by the respective GUI function right. Below API's are the exceptions from point 1 and 2 . Openness API will bypass the Function Right "Modify project via Openness API" and the API's are allowed irrespective of Function Right "Modify project via Openness API" set / not set.
  - project.ShowHwEditor(View.Network)
  - project.ShowHwEditor(View.Topology)
  - device.ShowInEditor(View.Device)
  - plcBlock.ShowInEditor()
  - plcType.ShowInEditor()
  - plcTagTable.ShowInEditor()
  - plcForceTable.ShowInEditor()
  - plcWatchTable.ShowInEditor()

5.3.15 Saving a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

To save a project

- Use the `Save()` method to save a project
- Use the `SaveAs()` method to save a project with a different name or in a different directory

Program code

Modify the following program code open and save a project:

```
public static void SaveProject(TiaPortal tiaPortal)
{
Project project = ...;
//Use the code in the try block to open and save a project
try
{
//please adapt the path and the extension apx to the installed version of TIA Portal
project = tiaPortal.Projects.Open(new FileInfo(@"Some\Path\MyProject.apx"));
//begin of code for further implementation
//...
//end of code'
project.Save();
}
//Use the code in the final block to close a project
finally
{
if (project != null)
project.Close();
}
}
```

Modify the following program code save a project with a different name or in a different location:

```
...
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
FileInfo fileInfoExistingProject = new FileInfo(@"D:\SampleProjects\SampleProject.apXX");
DirectoryInfo dirInfoSaveAsProject = new
DirectoryInfo(@"D:\SampleProjects\SampleProjectSaveAs");
Project sampleProject = portal.Projects.Open(fileInfoExistingProject );
sampleProject.SaveAs(dirInfoSaveAsProject);
...
```

5.3.16 Closing a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Program code

Modify the following program code to close a project:

```
public static void CloseProject(Project project)
{
project.Close();
}
```

5.3.17 Export/Import system diagnostics settings

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  SeeConnecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness application to export and import system diagnostics settings for a TIA Portal project. The format for export and import is .dat file.

The following service and class are used to export and import the setting for system diagnostics.

Name	Description
SystemdiagnosticsSettingsDataProvider	Service class to provide a method to export and import
SystemdiagnosticsSettingsExportImportResults	To return the result of Export or Import operation

The SystemdiagnosticsSettingsExportImportResultState ENUM has following values:

ENUM	Description	Value
SystemdiagnosticsSettingsExportImportResults	Returns the result whether the file is saved/ settings are changed successfully or not.	Error
		Success
		Warning

Program code: Export system diagnostics settings

Modify the following program code to export settings for system diagnostics:

```
public static void ExportSystemDiagnosticSettings(project tiaProject)
{
Siemens.Engineering.HW.Systemdiagnostics.Settings.SystemdiagnosticsSettingsDataProvider
settingsProvider =
tiaProject.GetService<Siemens.Engineering.HW.Systemdiagnostics.Settings.SystemdiagnosticsS
ettingsDataProvider>();
Siemens.Engineering.HW.Systemdiagnostics.Settings.SystemdiagnosticsSettingsExportImportRes
ult exportResult = settingsProvider.Export(new FileInfo(@"D:\Temp\SysDiagSettings.dat"));
Siemens.Engineering.HW.Systemdiagnostics.Settings.SystemdiagnosticsSettingsExportImportRes
ultState resultState = exportResult.State;
}
```

Program code: Import system diagnostics settings

Modify the following program code to import settings for system diagnostics:

```
public statc void ImportSystemDignosticSettings(project tiaProject)
{
Siemens.Engineering.HW.Systemdiagnostics.Settings.SystemdiagnosticsSettingsDataProvider
settingsProvider =
tiaProject.GetService<Siemens.Engineering.HW.Systemdiagnostics.Settings.SystemdiagnosticsS
ettingsDataProvider>();
Siemens.Engineering.HW.Systemdiagnostics.Settings.SystemdiagnosticsSettingsExportImportRes
ult importResult = settingsProvider.Import(new FileInfo(@"D:\Temp\SysDiagSettings.dat"));
Siemens.Engineering.HW.Systemdiagnostics.Settings.SystemdiagnosticsSettingsExportImportRes
ultState resultState = importResult.State;
}
```

5.4 Functions for accessing Teamcenter Gateway

5.4.1 Connecting to the Teamcenter Gateway

Requirement

- The TIA Portal application is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to establish a connection with Teamcenter via TcGateway.

Parameter

Parameter name	Data type	Mandatory	Description
userName	System.String	Yes	Specifies the username in Teamcenter
userPassword	System.SecureString	Yes	Specifies the password of user in Teamcenter
serverURL	System.String	Yes	Specifies the fully qualified URL of Teamcenter server
teamcenterInstance	System.String	Yes	Specifies the instance of Teamcenter server
userRole	System.String	No	Specifies the user role defined in Teamcenter.
userGroup	System.String	No	Specifies the group to which the user belongs in Teamcenter.

Note

A Siemens.Engineering exception of type TcGatewayException is thrown, if any of the above mandatory parameter is not passed.

A successful connection returns a encrypted TcGatewayConnectionInfo object. In case the connection is not successful, an engineering exception of type TcGatewayException is thrown containing the reason for failure.

Data type	Description
TcGatewayConnectionInfo	The TcGatewayConnection Info object has to be mandatorily passed in all the TcGateway openness APIs. It is used in other TcGateway Openness APIs to validate the source of the call and to validate the active Teamcenter session. The TcGatewayConnectionInfo provides the following connection details: • SessionToken (active connection session token) • Group (Teamcenter user group) • Role (Teamcenter user role)

Program code

```
//Connecting Teamcenter using TcGateway
public void function1()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal TIAInstance = ...;
TeamcenterConnectionProvider tcGatewayProvider =
TIAInstance.GetService<TeamcenterConnectionProvider>();
String userName = ...; //User name
SecureString password = ...; //Secure string. Client takes care of encrypting password and
openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...; //Group
String serverURL = ...; //Teamcenter server URL
String teamcenterInstance=....;//Instance
TcGatewayConnectionInfo connectionInfo = tcGatewayProvider.Connect(userName, password,
userGroup, userRole, serverURL,teamcenterInstance);
string group = connectionInfo.Group;
string role = connectionInfo.Role;
}
}
```

5.4.2 Disconnecting to the Teamcenter Gateway

Requirement

- The TIA Portal application is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to disconnect the connection between the Teamcenter server and TIA Portal using TcGateway.

Parameter

Parameter	Data type	Mandatory	Description
tcGatewayConnectionInfo	TcGatewayConnectionInfo	Yes	Returns the TcGatewayConnectionInfo object during Connect / ConnectSSO API call.

Program code

```
public void function2()
{
//Program code: Disconnect Teamcenter using TcGateway
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal TIAInstance = ...;
TcGatewayConnectionProvider tcGatewayProvider =
TIAInstance.GetService<TeamcenterConnectionProvider>();
String userName = ...;//User name
SecureString password = ...;//Secure string. Client takes care of encrypting password and
openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = ...;//Teamcenter server URL
String teamcenterInstance = ...;//Instance
TcGatewayConnectionInfo tcgConnectionInfo = tcGatewayProvider.Connect(userName,
password,serverURL,teamcenterInstance,userRole, userGroup);
tcGatewayProvider.Disconnect(tcgConnectionInfo);
}
}
```

See also

Connecting to the Teamcenter Gateway (Page 160)

5.4.3 Connecting to the ConnectSSO

Requirement

- The TIA Portal application is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to call the ConnectSSO() if an active SSO session is available with below required parameters.

Parameter	Data type	Mandatory	Description
hostURL	System.String	Yes	Specifies the fully qualified Teamcenter server host with port number.
instance	System.String	Yes	Specifies the Teamcenter server instance.

Parameter	Data type	Mandatory	Description
loginURL	System.String	Yes	Specifies the URL for the Login Service configured as part of Teamcenter Security Services.
applicationID	System.String	Yes	Specifies the Application ID configured in Teamcenter Application registry via Identity service

A successful connection returns an encrypted TcGatewayConnectionInfo object. In case the connection is not successful, an appropriate exception is thrown.

Note

If no active SSO session is available, you will be prompted to enter Teamcenter credentials in browser to create SSO session. The API supports secure card based login via PKI.

Program code

```
//Program code: ConnectSSO using TcGatewayConnectionInfo
public void function3()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal TIAInstance = ...;
TeamcenterConnectionProvider tcGatewayProvider =
TIAInstance.GetService<TeamcenterConnectionProvider>();
String hostUrl = ...;//Fully qualified Teamcenter server host with port number
String teamcenterInstance=...;//Instance
String loginURL = ....;//Teamcenter SSO login Service url
String applicationId = ...;//Teamcenter SSO application
TcGatewayConnectionInfo connectionInfo =
tcGatewayProvider.ConnectSSO(hostUrl,teamcenterInstance,loginURL,applicationId);
string group = connectionInfo.Group;
string role = connectionInfo.Role;
}
}
```

5.4.4 Saving Project/Global library in Teamcenter

Requirement

- The TIA Portal application is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to save a project/global library via the TcGatewayWorkflowProvider service. This service will be available at Project / Global Library composition.

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo during connection to Teamcenter. The tcGatewayConnectionInfo is passed to match with the active connection's tcGatewayConnectionInfo .
localCacheOption	LocalCacheOption(Enum)	Yes	TcGateway LocalCacheOption to overwrite the TIA project/ Global library in Teamcenter cache.

.

Datatype	Description
Siemens.Engineering.TeamcenterGateway.ItemInfo	The API returns the following ItemInfo object after the project or global library is successfully saved to the Teamcenter: • ItemId [System.String] • RevisionId [System.String] • ItemName [System.String] • ItemType [Siemens.Engineering.TeamcenterGateway.ItemType(Enum)]

Program code

```
//Program code: Saving Project/gobal library in Teamcenter using Save()
public void function4()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider= tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = "name";//User name
SecureString password = ...;//Secure string. Client takes care of encrypting password and
openness framework decrypts passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = ...;//Teamcenter server URL
String teamcenterInstance = ...;//Instance
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL,teamcenterInstance);
tcGatewayLockProvider.CheckoutDataset(connectionInfo ,"000495","A",DatasetType.T4TiaProjec
tDataset,"Project30");
ProjectComposition projectComposition = tiaPortal.Projects;
String projectPath = ...; // Project path for the checkout dataset in above step or download
project using TcGateway Load API Project
Project tcgProject = projectComposition.Open(new FileInfo(projectPath));
tcgProject.DeviceGroups.Create("Group_1");
tcgProject.Save(); //This is used to save project locally in TC cache. (optional)
TcGatewayWorkflowProvider tcgWorkflowProvider =
tcgProject.GetService<TcGatewayWorkflowProvider>();
ItemInfo response = tcgWorkflowProvider.Save(connectionInfo , LocalCacheOption.Overwrite);
//OR
ItemInfo response = tcgWorkflowProvider.Save(connectionInfo ,
LocalCacheOption.DoNotOverwrite);
tcGatewayLockProvider.CheckinDataset(connectionInfo ,"000495","A",DatasetType.T4TiaProject
Dataset,"Project30");
}
catch(TcGatewayException)
{
// other operations
}
}
}
```

5.4.5 Saving Project/Global library with proxy object

Requirement

- The TIA Portal application is connected to the TIA Portal Openness
  See Connection to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to archive the Project / Global library and then save the archive object to the Teamcenter dataset along with proxy objects (e.g. Plc, Program blocks, UDTs, other supported types).

Once the project / Global Library is successfully saved to Teamcenter, the API return ItemInfo object containing ItemId, RevisionId, ItemName and ItemType details.

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	TcGatewayConnectionInfo	Yes	Returns the tcGateway-ConnectionInfo during connection to Team-center. The tcGateway-ConnectionInfo has passed has to match with the active connec-tion's tcGatewayCon-nectionInfo
localCacheOption	LocalCacheOption(Enum)	Yes	TcGateway LocalCa-cheOption to overwrite TIA project/ Global li-brary in TC cache.

Program code: Saving project/global library in Teamcenter using SaveWithProxyObject( )

```
//Program code: Saving project/ global library in Teamcenter using SaveWithProxyObject()
public void function5()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = string.Empty;//User name
SecureString password = new SecureString();//Secure string. Client takes care of encrypting
password and openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = string.Empty;//Teamcenter server URL
String teamcenterInstance = string.Empty;//Instance
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
tcGatewayLockProvider.CheckoutDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");
ProjectComposition projectComposition = tiaPortal.Projects;
String projectPath = string.Empty; // Project path for the checkout dataset in above step or
download project using TcGateway Load API
Project tcgProject = projectComposition.Open(new FileInfo(projectPath));
TcGatewayWorkflowProvider tcgWorkflowProvider =
tcgProject.GetService<TcGatewayWorkflowProvider>();
ItemInfo response = tcgWorkflowProvider.SaveWithProxyObject(connectionInfo,
LocalCacheOption.Overwrite);
//OR
ItemInfo response = tcgWorkflowProvider.SaveWithProxyObject(connectionInfo,
LocalCacheOption.DoNotOverwrite);
tcGatewayLockProvider.CheckinDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");
}
catch (TcGatewayException)
{
// other operations
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

5.4.6 Saving Project/Global library with new item

Requirement

- Connecting to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo for successful connection to the Teamcenter. The tcGatewayConnectionInfo has passed and must match with the active connection's tcGatewayConnectionInfo .
itemId	System.String	Yes	Specifies the item id to which project needs to be saved.
revisionId	System.String	Yes	Specifies the revision id to which project needs to be saved.
localCacheOption	Siemens.Engineering.TeamcenterGateway.LocalCacheOption(Enum)	Yes	Specifies the overwrite local cache project contents if local cache or dataset version of project mismatches with server project version

Program code

```
//Program code: Saving project/global library in Teamcenter using SaveToItem( )
public void function6()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = string.Empty;//User name
SecureString password = new SecureString();//Secure string. Client takes care of encrypting
password and openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = string.Empty;//Teamcenter server URL
String teamcenterInstance = string.Empty;//Instance
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
tcGatewayLockProvider.CheckoutDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");
ProjectComposition projectComposition = tiaPortal.Projects;
String projectPath = string.Empty; // Project path for the checkout dataset in above step or
download project using TcGateway Download API
Project tcgProject = projectComposition.Open(new FileInfo(projectPath));
TcGatewayWorkflowProvider tcgWorkflowProvider =
tcgProject.GetService<TcGatewayWorkflowProvider>();
ItemInfo response = tcgWorkflowProvider.SaveToItem(connectionInfo, "000495", "A",
LocalCacheOption.Overwrite);
tcGatewayLockProvider.CheckinDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");
}
catch (TcGatewayException)
{
// other operations
}
}
}
```

5.4.7 Accessing custom attributes in Teamcenter

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to get and set the custom attributes value in the Teamcenter Gateway.

Parameter

The Teamcenter itemtype refers to T4TiaLibrary,T4TiaProject and subtypes which are present in Teamcenter Datamodel for which custom attributes are mapped in Teamcenter.

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo during connection to Teamcenter. The tcGatewayConnectionInfo is passed and has to match with the active connection's tcGatewayConnectionInfo .
itemType	System.String	Yes	Specifies the Teamcenter itemType

After successful execution, the API returns collection of custom attributes if available in the Teamcenter for given itemtype.

Data type	Description
IList<Siemens.Engineering.TeamcenterGateway.TeamcenterProperty>	Specifies the list of Teamcenter custom property for item or revision. • Name [System.String] • MappingLevel [System.int] • ListOfValueInfo [Siemens.Engineering.TeamcenterGateway.TcPropertyListOfValueInfo] – LowerLimit [System.String] – UpperLimit [System.String] – Values [System.Collections.Generic.IEnumerable<System.String>] • DataType [Siemens.Engineering.TeamcenterGateway.MappedCustomAttributeType(Enum)]

The Call SetValue API to set value on custom attributes fetched using GetTeamcenterCustomAttributes API along with pass error callback delegate.

Parameter name	Data type	Is Mandatory	Description
value	System.String	Yes	Specifies the value for custom attribute
errorCallback	Siemens.Engineering.TeamcenterGateway.ErrorCallback	Yes	Delegate to handle error which are raised during API invocation/ actions.

Program code

```
public void function7()
{
using Siemens.Engineering.TeamcenterGateway;
{
using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
{
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = string.Empty;//User name
SecureString password = new SecureString();//Secure string. Client takes care of encrypting
password and openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = string.Empty;//Teamcenter server URL
String teamcenterInstance = string.Empty;//Instance
ItemDetailsDelegate itemDetailsDelegate = ItemDetailsForCreation;
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
var project = tiaPortal.Projects.Open(new FileInfo("D:\\UserData\\Documents\\Automation\
\Project30\\Project30.ap18"));
var tcGatewayWorkflowProvider = project.GetService<TcGatewayWorkflowProvider>();
//OR
var library = tiaPortal.GlobalLibraries.Open(new
FileInfo(@"~\Documents\Automation\Library7\Library7.al18"), OpenMode.ReadWrite);
var tcGatewayWorkflowProvider = library.GetService<TcGatewayWorkflowProvider>();
//Get custom attributes.
var attributes = tcGatewayWorkflowProvider.GetTeamcenterCustomAttributes(connectionInfo,
"T4TiaProject");
ErrorCallback errorCallback = PropertyValueSettingErrorCallback;
foreach (var arr in attributes)
{
if (arr.ListOfValueInfo != null)
{
foreach (var l in arr.ListOfValueInfo.Values)
{
Console.WriteLine(l);
}
}
switch (arr.DataType)
{
case MappedCustomAttributeType.String:
arr.SetValue("23232323232", errorCallback);
break;
case MappedCustomAttributeType.Char:
arr.SetValue("d", errorCallback);
break;
case MappedCustomAttributeType.Integer:
arr.SetValue("232", errorCallback);
break;
case MappedCustomAttributeType.arr.SetValue(23.4f.ToString(), errorCallback);
break;
}
}
```

```
}
catch (TcGatewayException)
{
// other operations
}
}
}
}
```

5.4.8 Saving Project to Teamcenter as new item

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Hotspot-TextConnecting to the TIA Portal (Page 82)
- A Project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to save a TIA Portal project in Teamcenter as a new item. The API returns itemInfo object containing the ItemId, RevisionId, ItemType, ItemName after successful creation of item/revision.

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo during connection to Teamcenter. The tcGatewayConnectionInfo has to match with the active connection's tcGatewayConnectionInfo .
itemDetailsDelegate	delegate	Yes	This delegate is defined by caller of API which contains information about the item to be created in Teamcenter: • ItemId [System.String] • RevisionId [System.String] • ItemName [System.String](Required) • Comments [System.String] • TeamcenterFolder [System.String] • TeamcenterProject [System.String[]] • TeamcenterItemType [System.String](Required)
customAttributes	IEnumerable<Siemens.Engineering.TeamcenterGateway.TeamcenterProperty>	No	Specifies the Item/Revision custom attributes which are saved along with Project / Global Library.

Program code

```
private static void ItemDetailsForCreation(ItemDetails itemDetails)
{
itemDetails.ItemId = "...";
itemDetails.ItemName = "Proj_Object";
itemDetails.RevisionId = "...";
itemDetails.TeamcenterItemType = "T4TiaProject";
itemDetails.Comment = "Prooject Comments.";
itemDetails.TeamcenterFolder = "Home\\TIA_Project\\ConveyourBelt";
itemDetails.TeamcenterProject = new string[] { "Project_TIA_1", "Project_TIA_2" };
}
```

```
private static void PropertyValueSettingErrorCallback(string errorMessage)
{
Console.WriteLine(errorMessage);
}
```

```
//Program code: Saving project to Teamcenter as new item
private void functionSavingProjectToTeamcenterAsNewItem()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = string.Empty;//User name
SecureString password = new SecureString();//Secure string. Client takes care of encrypting
password and openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = string.Empty;//Group
String serverURL = string.Empty;//Teamcenter server URL
String teamcenterInstance = string.Empty;//Instance
ItemDetailsDelegate itemDetailsDelegate = ItemDetailsForCreation;
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
var project = tiaPortal.Projects.Open(new FileInfo("D:\\UserData\\Documents\\Automation\
\Project30\\Project30.ap18"));
var tcGatewayWorkflowProvider = project.GetService<TcGatewayWorkflowProvider>();
//OR
var library = tiaPortal.GlobalLibraries.Open(new
FileInfo(@"~\Documents\Automation\Library7\Library7.al18"), OpenMode.ReadWrite);
var tcGatewayWorkflowProvider = library.GetService<TcGatewayWorkflowProvider>();
//Save item in Teamcenter with custom attributes.
var attributes = tcGatewayWorkflowProvider.GetTeamcenterCustomAttributes(connectionInfo,
"T4TiaProject");
ErrorCallback errorCallback = PropertyValueSettingErrorCallback;
foreach (var arr in attributes)
{
if (arr.ListOfValueInfo != null)
{
//Err:
//Invalid arr.ListOfValueInfo element
foreach (var l in arr.ListOfValueInfo?.Values)
{
Console.WriteLine(l);
}
}
switch (arr.DataType)
{
case MappedCustomAttributeType.String:
arr.SetValue("23232323232", errorCallback);
break;
case MappedCustomAttributeType.Char:
arr.SetValue("d", errorCallback);
break;
case MappedCustomAttributeType.Integer:
arr.SetValue("232", errorCallback);
break;
case MappedCustomAttributeType.Float:
arr.SetValue(23.4f.ToString(), errorCallback);
```

```
break;
}
var createdItem = tcGatewayWorkflowProvider.SaveAsNewItem(connectionInfo,
itemDetailsDelegate, attributes);
//OR
var createdItem = tcGatewayWorkflowProvider.SaveAsNewItem(connectionInfo,
itemDetailsDelegate, null);//< --optional
var itemId = createdItem.ItemId;
var revisionId = createdItem.RevisionId;
var itemName = createdItem.ItemName;
var type = createdItem.ItemType;
//Checkout created dataset to edit in TIA portal
tcGatewayLockProvider.CheckoutDataset(connectionInfo, itemId, revisionId,
DatasetType.T4TiaProjectDataset, itemName);
//Check in dataset after changes to project are completed
tcGatewayLockProvider.CheckinDataset(connectionInfo, itemId, revisionId,
DatasetType.T4TiaProjectDataset, itemName);
}
}
catch (TcGatewayException)
{
// other operations
}
}
}
```

5.4.9 Saving Project as new item with proxy object

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to save the TIA Portal project and Global library to the Team Center along with proxy object. The API returns itemInfo object details such as ItemId, RevisionId, ItemType, ItemName after successful creation of new item.

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo during connection to Teamcenter. The tcGatewayConnectionInfo is passed and should match with the active connection's tcGatewayConnectionInfo .
itemDetailsDelegate	delegate	Yes	This delegate is defined by caller of API which has information about item to be created in Teamcenter: • ItemId [System.String] • RevisionId [System.String] • ItemName [System.String](Required) • Comments [System.String] • TeamcenterFolder [System.String] • TeamcenterProject [System.String[]] • TeamcenterItemType [System.String](Required)
customAttributes	IEnumerable<Siemens.Engineering.TeamcenterGateway.TeamcenterProperty>	No	Specifies the Item/Revision custom attributes saved along with Project / Global Library.

Program code

```
private static void ItemDetailsForCreation(ItemDetails itemDetails)
{
itemDetails.ItemId = "...";
itemDetails.ItemName = "Proj_Object";
itemDetails.RevisionId = "...";
itemDetails.TeamcenterItemType = "T4TiaProject";
itemDetails.Comment = "Project Comments.";
itemDetails.TeamcenterFolder = "Home\\TIA_Project\\ConveyourBelt";
itemDetails.TeamcenterProject = new string[] { "Project_TIA_1", "Project_TIA_2" };
}
```

```
private static void PropertyValueSettingErrorCallback(string errorMessage)
{
Console.WriteLine(errorMessage);
}
```

```
//Program code: Saving project/library in Teamcenter as new item with proxy object
private void function SavingProjectAsNewItemWithProxyObject()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = string.Empty;//User name
SecureString password = new SecureString();//Secure string. Client takes care of encrypting
password and openness framework decrypts and passes the decrypted value to the action
String userRole = string.Empty; ;//Role
String userGroup = string.Empty;//Group
String serverURL = string.Empty;//Teamcenter server URL
String teamcenterInstance = string.Empty;//Instance
ItemDetailsDelegate itemDetailsDelegate = ItemDetailsForCreation;
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
var project = tiaPortal.Projects.Open(new FileInfo("D:\\UserData\\Documents\\Automation\
\Project30\\Project30.ap18"));
var tcGatewayWorkflowProvider = project.GetService<TcGatewayWorkflowProvider>();
//OR
var library = tiaPortal.GlobalLibraries.Open(new
FileInfo(@"~\Documents\Automation\Library7\Library7.al18"), OpenMode.ReadWrite);
var tcGatewayWorkflowProvider = library.GetService<TcGatewayWorkflowProvider>();
//Save item in Teamcenter with custom attributes.
var attributes = tcGatewayWorkflowProvider.GetTeamcenterCustomAttributes(connectionInfo,
"T4TiaProject");
ErrorCallback errorCallback = PropertyValueSettingErrorCallback;
foreach (var arr in attributes)
{
if (arr.ListOfValueInfo != null)
{
foreach (var l in arr.ListOfValueInfo?.Values)
{
Console.WriteLine(l);
}
}
switch (arr.DataType)
{
case MappedCustomAttributeType.String:
arr.SetValue("23232323232", errorCallback);
break;
case MappedCustomAttributeType.Char:
arr.SetValue("d", errorCallback);
break;
case MappedCustomAttributeType.Integer:
arr.SetValue("232", errorCallback);
break;
case MappedCustomAttributeType.Float:
arr.SetValue(23.4f.ToString(), errorCallback);
break;
}
```

```
var createdItem = tcGatewayWorkflowProvider.SaveAsNewItemWithProxyObject(connectionInfo,
itemDetailsDelegate, attributes);
//OR
var createdItem = tcGatewayWorkflowProvider.SaveAsNewItem(connectionInfo,
itemDetailsDelegate, null); //< --optional
var itemId = createdItem.ItemId;
var revisionId = createdItem.RevisionId;
var itemName = createdItem.ItemName;
var type = createdItem.ItemType;
//Checkout created dataset to edit in TIA portal
tcGatewayLockProvider.CheckoutDataset(connectionInfo, itemId, revisionId,
DatasetType.T4TiaProjectDataset, itemName);
//Check in dataset after changes to project are complete
tcGatewayLockProvider.CheckinDataset(connectionInfo, itemId, revisionId,
DatasetType.T4TiaProjectDataset, itemName);
}
}
catch (TcGatewayException)
{
// other operations
}
}
}
```

5.4.10 Saving Project to Teamcenter as new revision

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to save the TIA Portal project and Global library in Teamcenter as new revision. The API returns the ItemInfo object such as ItemId, RevisionId, ItemName and ItemType details after successful creation of new revision.

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo during connection to Teamcenter. The tcGatewayConnectionInfo has passed to match with the active connection's tcGatewayConnectionInfo
revisionDetailsDelegate	RevisionDetailsDelegate	No	This delegate is defined by caller of API which has information about revision to be created in Teamcenter
customAttributes	IEnumerable<Siemens.Engineering.TeamcenterGateway.TeamcenterProperty>	No	Specifies the Item/Revision custom attributes saved along with Project / Global Library.

Program code

```
//Program code: Saving project/library in Teamcenter as new revision
public static void RevisionDetailsForCreation(RevisionDetails revisionDetails)
{
revisionDetails.RevisionId = "B";
revisionDetails.Comment = "Revised project";
}
```

```
private void SavingProjectToTeamcenterAsNewRevision()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = string.Empty;//User name
SecureString password = new SecureString();//Secure string. Client takes care of encrypting
password and openness framework decrypts and passes the decrypted value to the action;
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = string.Empty;//Teamcenter server URL
String teamcenterInstance = string.Empty;//InstanceRevisionDetailsDelegate
revisionDetailsDelegate = RevisionDetailsForCreation;
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
var project = tiaPortal.Projects.Open(new FileInfo("D:\\UserData\\Documents\\Automation\
\Project30\\Project30.ap18"));
var tcGatewayWorkflowProvider = project.GetService<TcGatewayWorkflowProvider>();
ItemDetailsDelegate itemDetailsDelegate = ItemDetailsForCreation;
void ItemDetailsForCreation(ItemDetails itemDetails)
{
itemDetails.ItemId = "...";
itemDetails.ItemName = "Project30";
itemDetails.RevisionId = "...";
itemDetails.TeamcenterItemType = "T4TiaProject";
itemDetails.Comment = "Project Comments.";
}
var saveAsNewItemResult = tcGatewayWorkflowProvider.SaveAsNewItem(connectionInfo,
itemDetailsDelegate, null);
//Checkout created dataset to edit in TIA portal
tcGatewayLockProvider.CheckoutDataset(connectionInfo, "000495",
"A"),DatasetType.T4TiaProjectDataset, "Project30");
var saveAsNewRevisionResult = tcGatewayWorkflowProvider.SaveAsNewRevision(connectionInfo,
revisionDetailsDelegate, null);
//Check in dataset after changes to project are complete
tcGatewayLockProvider.CheckinDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");
//Checkout revised project/library
tcGatewayLockProvider.CheckoutDataset(connectionInfo, "000495", "B",
DatasetType.T4TiaProjectDataset, "Project30");
//Check in revised dataset after changes to project are complete
tcGatewayLockProvider.CheckinDataset(connectionInfo, "000495", "B",
DatasetType.T4TiaProjectDataset, "Project30");
}
catch (TcGatewayException)
{
// other operations
}
}
}
```

5.4.11 Downloading Project/Global library from Teamcenter

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to download and open the project and library from Teamcenter. The API returns starter file path to open the project or library in the TIA Portal.

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo returned during connection to Teamcenter. The tcGatewayConnectionInfo has passed has to match with the active connection's tcGatewayConnectionInfo .
itemID	System.String	Yes	Specifies the Item ID of the TIA Project / Library.
revisionID	System.String	Yes	Specifies the Revision ID of the TIA Project / Library.
itemType	Siemens.Engineering.TeamcenterGateway.ItemType(Enum)	Yes	Item type of Project/Library in Teamcenter.
localCacheOption	Siemens.Engineering.TeamcenterGateway.LocalCacheOption(Enum	Yes	Overwrite the contents of TcCache incase a version of the Project / Global Library being downloaded is already available. True means overwrite the content. Incase of False a exception is thrown if there is a version mismatch between TcCache and Teamcenter or if the contents have been modified.

Note

If any of the mandatory parameter is not passed, a Siemens.Engineering exception of type TcGatewayException is thrown.

Program code

```
//Program code: Download project/library in Teamcenter
private void DownloadProjectLibrary()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider= tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
String userName = ...;//User name
SecureString password = ...;//Secure string. Client takes care of encrypting password and
openness framework decrypts and passes the decrypted value to the action;
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = ...;//Teamcenter server URL
String teamcenterInstance = ...;//Instance
try
{
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
tcGatewayLockProvider.CheckoutDataset(connectionInfo ,"000495","A",DatasetType.T4TiaProjec
tDataset,"Project30");
TcGatewaySearchAndDownloadProvider downloadFromTcProvider =
tiaPortal.GetService<TcGatewaySearchAndDownloadProvider>();
FileInfo TIAObjectStarterFilePath =
downloadFromTcProvider.Download(connectionInfo,"000495", "A", ItemType.Project,
LocalCacheOption.Overwrite);
}
catch(TcGatewayException)
{
// other operations
}
var project = tiaPortal.Projects.Open(TIAObjectStarterFilePath);
//OR
var library=tiaPortal.GlobalLibraries.Open(TIAObjectStarterFilePath, OpenMode.ReadWrite);
//User can work on the Project.
project.DeviceGroups.Create("Group_3");
project.Save();
}
}
```

5.4.12 Searching Project / Global library from Teamcenter

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A Project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to search project and global library in Teamcenter. The Search API returns a list of TIA Portal objects from Teamcenter with search result containing the details such as ItemID and List of related RevisionIds.

Parameter

Parameter	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the Teamcenter connectionInfo after sucessful connection.
type	Siemens.Engineering.TeamcenterGateway.ItemType(Enum)	Yes	Specifies the Item type of project / library in Teamcenter.
TIAObjectName	System.String	No	Specifies the name of the TIA Project / Library. It provides a complete name or with wild card characters. [User should provide value in at least one of the fields; Item ID, Item Name, TIAObjectName]
itemID	System.String	No	Specifies the item ID of the TIA Project / Library. It Provides a complete Item Id or with wild card characters. [User should provide value in at least one of the fields; Item ID, Item Name, TIAObjectName]
itemName	System.String	No	Specifies the item name of the TIA Project / Library. Provide a complete Item Name or with wild card characters. [User should provide value in at least one of the fields; Item ID, Item Name, TIAObjectName]
revisionID	System.String	No	Specifies the revision ID of the TIA Project / Library. User shall either enter a complete revision ID or with wild card characters.

Note

A Siemens.Engineering exception of type TcGatewayException is thrown if the required mandatory parameters are not passed to search project / library in Teamcenter.

Program code

```
//Program code: Search project/library in Teamcenter
private void function9()
{
using Siemens.Engineering.TeamcenterGateway;
using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
{
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
String userName = ...;//User name
SecureString password = ...;//Secure string. Client takes care of encrypting password and
openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = ...;//Teamcenter server URL
String teamcenterInstance =...;//Instance
try
{
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
TcGatewaySearchAndDownloadProvider searchAndLoadProvider =
tiaPortal.GetService<TcGatewaySearchAndDownloadProvider>();
var items = searchAndLoadProvider.Search(connectionInfo, ItemType.Project, "P*", "", "",
"");
}
catch (TcGatewayException)
{
// other operations
foreach (var item in items)
{
var itemId = item.ItemId;
var revisionIdList = item.RevisionId;
foreach (var revision in revisionIdList)
{
Console.WriteLine(itemId + "-" + revision);
}
}
}
}
}
```

5.4.13 Checkout Project/Global library dataset in Teamcenter

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Opening a project (Page 128)
- A Project is open
  See Opening a project (Page 82)

Application

You can use the TIA Portal Openness to perform checkout action on TIA Portal project and Global library in Teamcenter dataset.

Parameter

Parameter	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo during connection to Teamcenter. The tcGatewayConnectionInfo must passed to match with the active connection's tcGatewayConnectionInfo .
itemID	System.String	Yes	Specifies the item ID of the TIA Project / Library in Teamcenter
revisionID	System.String	Yes	Specifies the revision ID of the TIA Project / Library in Teamcenter to which the dataset is linked.
datasetType	Siemens.Engineering.TeamcenterGateway.DatasetType	Yes	Specifies the Teamcenter dataset type corresponding to either project / library to be checked out. Eg: T4TiaProjectDataset / T4TiaLibraryDataset.
datasetName	System.String	Yes	Specifies the Teamcenter dataset name corresponding to either project / library to be checked out.

Note

A Siemens.Engineering exception of type TcGatewayException is thrown if the required mandatory parameters are not passed to perform checkout action on project / library in Teamcenter.

Program code: Checkout dataset in Teamcenter

```
//Program code: Checkout project/global library in Teamcenter dataset
private void function10()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = ...;//User name
SecureString password = ...;//Secure string. Client takes care of encrypting password and
openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = ...;//Teamcenter server URL
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
tcGatewayLockProvider.CheckoutDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");
}
catch (TcGatewayException)
{
//User can perform saving of Changes of TIA Project to the checked out dataset or any other
operations
}
}
}
```

5.4.14 Checkin Project/Global library dataset in Teamcenter

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to check in action on TIA Portal project and Global library in Teamcenter dataset.

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Return the tcGatewayConnectionInfo during connection to Teamcenter. The tcGatewayConnectionInfo must be passed to match with the active connection's tcGatewayConnectionInfo .
itemID	System.String	Yes	Specifies the item ID of the TIA Portal Project / Library in Teamcenter
revisionID	System.String	Yes	Specifies the revision ID of the TIA Portal Project / Library in Teamcenter to which the dataset is linked.
datasetType	Siemens.Engineering.TeamcenterGateway.DatasetType	Yes	Specifies the Teamcenter dataset type corresponding to either project / library to be checked in. Eg: T4TiaProjectDataset / T4TiaLibraryDataset.
datasetName	System.String	Yes	Specifies the Teamcenter dataset name corresponding to either project / library to be checkedin.

Note

A Siemens.Engineering exception of type TcGatewayException is thrown if any of the above mandatory parameters is not passed for checkin of project / library to the Teamcenter dataset.

Program code

```
//Program code: Checkin project/global library in Teamcenter dataset
private void function11()
{
using Siemens.Engineering.TeamcenterGateway;
{
using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
{
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = ...;//User name
SecureString password = ...;//Secure string. Client takes care of encrypting password and
openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = ...;//Teamcenter server URL
String teamcenterInstance =....;//Instance
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
tcGatewayLockProvider.CheckoutDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");//User can perform saving of Changes of TIA
Project to the checkedout dataset using TcGateway Save APIs.
tcGatewayLockProvider.CheckinDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");//Checkin to release the lock on dataset
}
catch (TcGatewayException)
{
// other operations
}
}
}
}
```

5.4.15 Cancel checkout dataset in Teamcenter

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A Project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to cancel checkout action on TIA Portal project and Global library dataset in Teamcenter. You can also use the CancelCheckoutDataset() to release the lock on dataset in Teamcenter.

Parameter

Parameter name	Data type	Is Mandatory	Description
tcGatewayConnectionInfo	Siemens.Engineering.TeamcenterGateway.TcGatewayConnectionInfo	Yes	Returns the tcGatewayConnectionInfo during connection to Teamcenter. The tcGatewayConnectionInfo must be passed to match with the active connection's tcGatewayConnectionInfo .
itemID	System.String	Yes	Specifies the item ID of the TIA Project / Library in Teamcenter
revisionID	System.String	Yes	Specifies the revision ID of the TIA Project / Library in Teamcenter to which the dataset is linked.
datasetType	Siemens.Engineering.TeamcenterGateway.DatasetType	Yes	Specifies the Teamcenter dataset type corresponding to either project / library to be checked out. Eg: T4TiaProjectDataset / T4TiaLibraryDataset.
datasetName	System.String	Yes	Specifies the Teamcenter dataset name corresponding to either project / library to be checked out.

Note

A Siemens.Engineering exception of type TcGatewayException is thrown if any of the above mandatory parameters is not passed to cancel checkout action on project / library in Teamcenter.

Program code

```
//Program code: Cancel checkout datatset in Teamcenter
private void function12()
{
using Siemens.Engineering.TeamcenterGateway;
{
TiaPortal tiaPortal = ...;
var tcGatewayConnectionProvider = tiaPortal.GetService<TeamcenterConnectionProvider>();
var tcGatewayLockProvider = tiaPortal.GetService<TcGatewayLockProvider>();
try
{
String userName = ...;//User name
SecureString password = ...;//Secure string. Client takes care of encrypting password and
openness framework decrypts and passes the decrypted value to the action
String userRole = ...;//Role
String userGroup = ...;//Group
String serverURL = ...;//Teamcenter server URL
String teamcenterInstance = ...;//Instance
TcGatewayConnectionInfo connectionInfo = tcGatewayConnectionProvider.Connect(userName,
password, userGroup, userRole, serverURL, teamcenterInstance);
//User can perform saving of Changes of TIA Project to the checked out dataset using
TcGateway Save APIs.
tcGatewayLockProvider.CheckoutDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");
//Cancel checkout on dataset to release the lock on dataset and undo the changes done on
dataset
tcGatewayLockProvider.CancelCheckoutDataset(connectionInfo, "000495", "A",
DatasetType.T4TiaProjectDataset, "Project30");
}
catch (TcGatewayException)
{
// other operations
}
}
}
```

5.5 Functions for Connections

5.5.1 Configurable attributes of a port-to-port connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Opening a project (Page 128)
- A project is open.
  See Opening a project (Page 128)

Application

The attributes of a port interconnection are located at the port device item. The read and write access of attributes via TIA Portal Openness is the same as at the UI.

Port interface settings

The following attributes are provided for port interface settings:

Attribute name	Data type	Writeable	Access	Description
`MediumAttachmentType`	MediumAttachment-Type	r/o	Dynamic at-tribute	-
`CableName`	CableName	r/w	Dynamic at-tribute	-
`AlternativePartnerPorts`	Boolean	r/w	Dynamic at-tribute	Only available if toolchanger functionality is supported, e.g. at CPU1516.
`SignalDelaySelection`	SignalDelaySelection	r/w	Dynamic at-tribute	-
`CableLength`	CableLength	r/w	Dynamic at-tribute	-
`SignalDelayTime`	Double	r/w	Dynamic at-tribute	-

The following ENUM values are provided for the attribute `MediumAttachmentType`:

Value	Description
`MediumAttachmentType.None`	Attachment type cannot be determined.
`MediumAttachmentType.Copper`	Attachment type is copper.
`MediumAttachmentType.FibreOptic`	Attachment type is fiber optic.

The following ENUM values are provided for the attribute `CableName`:

Value	Description
`CableName.None`	No cable name is specified
`CableName.FO_Standard_Cable_9`	FO standard cable GP (9 µm)
`CableName.Flexible_FO_Cable_9`	Flexible FO cable (9 µm)
`CableName.FO_Standard_Cable_GP_50`	FO standard cable GP (50 µm)
`CableName.FO_Trailing_Cable_GP`	FO trailing cable / GP
`CableName.FO_Ground_Cable`	FO ground cable
`CableName.FO_Standard_Cable_62_5`	FO standard cable (62.5 µm)
`CableName.Flexible_FO_Cable_62_5`	Flexible FO cable (62.5 µm)
`CableName.POF_Standard_Cable_GP`	POF standard cable GP
`CableName.POF_Trailing_Cable`	POF trailing cable
`CableName.PCF_Standard_Cable_GP`	PCF standard cable GP
`CableName.PCF_Trailing_Cable_GP`	PCF trailing cable / GP
`CableName.GI_POF_Standard_Cable`	GI-POF standard cable
`CableName.GI_POF_Trailing_Cable`	GI-POF trailing cable

Value	Description
`CableName.GI_PCF_Standard_Cable`	GI-PCF standard cable
`CableName.GI_PCF_Trailing_Cable`	GI-PCF trailing cable

The following ENUM values are provided for the attribute `SignalDelaySelection`:

Value	Description
`SignalDelaySelection.None`	-
`SignalDelaySelection.CableLength`	CableLength is used to define the signal delay.
`SignalDelaySelection.SignalDelayTime`	SignalDelayTime is used to define the signal delay.

The following ENUM values are provided for the attribute `CableLength`:

Value	Description
`CableLength.None`	Cable length is not specified.
`CableLength.Length20m`	Cable length is 20m.
`CableLength.Length50m`	Cable length is 50m.
`CableLength.Length100m`	Cable length is 100m.
`CableLength.Length1000m`	Cable length is 1000m.
`CableLength.Length3000m`	Cable length is 3000m.

Port options

The following attributes are provided for port options:

Attribute name	Data type	Writeable	Access
`PortActivation`	bool	r/w	Dynamic attribute
`TransmissionRateAndDuplex`	TransmissionRateAndDuplex	r/w	Dynamic attribute
`PortMonitoring`	bool	r/w	Dynamic attribute
`TransmissionRateAutoNegotiation`	bool	r/w	Dynamic attribute
`EndOfDetectionOfAccessibleDevices`	bool	r/w	Dynamic attribute
`EndOfTopologyDiscovery`	bool	r/w	Dynamic attribute
`EndOfSyncDomain`	bool	r/w	Dynamic attribute

The following ENUM values are provided for the attribute `TransmissionRateAndDuplex`:

Value	Description
`TransmissionRateAndDuplex.None`	-
`TransmissionRateAndDuplex.Automatic`	Automatic
`TransmissionRateAndDuplex.AUI10Mbps`	10 Mbps AUI
`TransmissionRateAndDuplex.TP10MbpsHalfDuplex`	TP 10 Mbps half duplex
`TransmissionRateAndDuplex.TP10MbpsFullDuplex`	TP 10 Mbps full duplex

Value	Description
`TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex`	async fiber 10Mbit/s half duplex mode
`TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex`	async fiber 10Mbit/s full duplex mode
`TransmissionRateAndDuplex.TP100MbpsHalfDuplex`	TP 100 Mbps half duplex
`TransmissionRateAndDuplex.TP100MbpsFullDuplex`	TP 100 Mbps full duplex
`TransmissionRateAndDuplex.FO100MbpsFullDuplex`	FO 100 Mbps full duplex
`TransmissionRateAndDuplex.X1000MbpsFullDuplex`	X1000 Mbps full Duplex
`TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD`	FO 1000 Mbps full duplex LD
`TransmissionRateAndDuplex.FO1000MbpsFullDuplex`	FO 1000 Mbps full Duplex
`TransmissionRateAndDuplex.TP1000MbpsFullDuplex`	TP 1000 Mbps full duplex
`TransmissionRateAndDuplex.FO10000MbpsFullDuplex`	FO 10000 Mbps full Duplex
`TransmissionRateAndDuplex.FO100MbpsFullDuplexLD`	FO 100 Mbps full duplex LD
`TransmissionRateAndDuplex.POFPCF100MbpsFullDuplex`	POF/PCF 100 Mbps full duplex

See also

Connecting to the TIA Portal (Page 82)

5.6 Functions on libraries

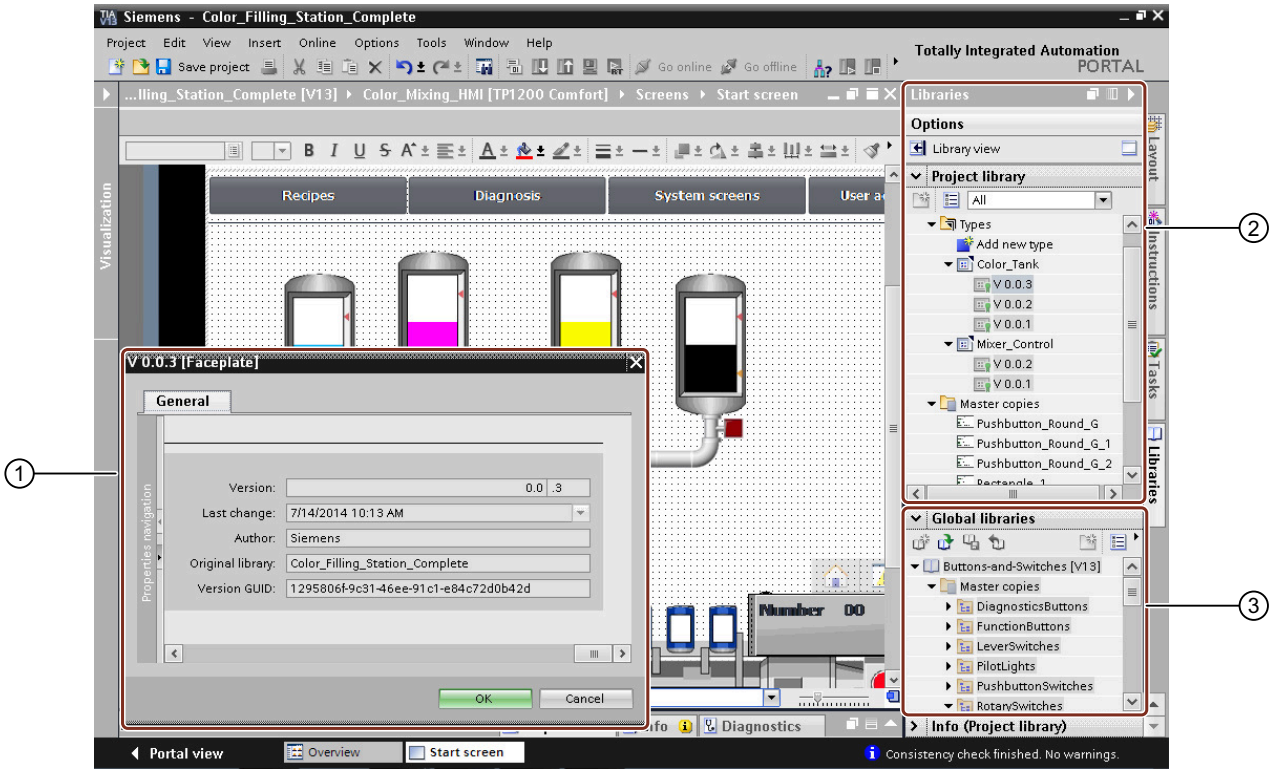
5.6.1 Functions for objects and instances

Accessing types and instances

You can use the TIA Portal Openness API interface to access types, type versions and master copies in the project library or global libraries. You can determine connections between type versions and instances. You can also update instances in the project and synchronize changes between a global library and the project library. The TIA Portal Openness API interface also supports the comparison of types versions and instances.

Functions for objects and instances

You have access to the following functions for types, type versions, master copies and instances with the TIA Portal Openness API interface:



① Display attributes of types, type versions, master copies and instances

② The following functions are available in the project library:

- Update instances of types
- Instantiating type versions in the project
- Navigate within the library group
- Delete groups, types, type versions and master copies

③ The following functions are available in the global library:

- Update instances of types
- Instantiate type version in the project
- Navigate within the library group

5.6.2 Accessing global libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)

Application

Three types of Global Libraries are existing.

- System global library (Siemens.Engineering.Library.SystemGlobalLibrary): These global libraries are included as part of a TIA Portal installation and use the .asx file extension. All system global libraries are read only.
- Corporate global library (Siemens.Engineering.Library.CorporateGlobalLibrary): These global libraries have been chosen by an administrator to be preloaded when TIA Portal is started. All corporate global libraries are read only.
- User global library (Siemens.Engineering.Library.UserGlobalLibrary): These global libraries have been created by users of TIA Portal. User global libraries can be opened either as read only mode or readwrite mode.
  If an user global library is already openned in a certain mode then the same user global library cannot be openned with another mode.
  User global libraries from previous versions can be opened only in read only mode.

Global Libraries opened using TIA Portal Openness will also be added to the TIA Portal UI's global library collection, and seen in the TIA Portal UI if the UI is present.

Program code: Available global libraries

Modify the following program code to get informations about all available global libraries:

```
TiaPortal tia = ...;
var availableLibraries = tia.GlobalLibraries.GetGlobalLibraryInfos();
foreach (GlobalLibraryInfo info in availableLibraries)
{
Console.WriteLine(" Library Name: {0}", info.Name);
Console.WriteLine(" Library Path: {0}", info.Path);
Console.WriteLine(" Library Type: {0}", info.LibraryType);
Console.WriteLine(" Library IsOpen: {0}", info.IsOpen);
}
```

Attributes of GlobalLibrary

Value	Data type	Description
Author	String	Author of the global library.
Comment	MultilingualText	Comment of the global library.
IsReadOnly	Boolean	True if the global library is read only.
IsModified	Boolean	True if the contents of the global library has been modified.
Name	String	Name of the global library.
Path	FileInfo	Path of the global library.

Attributes of GlobalLibraryInfo

Value	Return Type	Description
IsReadOnly	Boolean	True if the global library is read only.
IsOpen	Boolean	True if the global library is already open.
LibraryType	GlobalLibraryType	Type of the global library: • System: System global library • Corporate: Corporate global library • User: User global library
Name	String	Name of the global library.
Path	FileInfo	Path of the global library.

See also

Accessing folders in a library (Page 217)

5.6.3 Accessing library base type

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access the base type classes of the library type object. These all have a direct base type of Siemens.Engineering.Library.Types.LibraryType

You can have access to the below base type classes:

- Siemens.Engineering.Hmi.Faceplate.FaceplateLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.CScriptLibraryType
- Siemens.Engineering.Hmi.Screen.ScreenLibraryType
- Siemens.Engineering.Hmi.Screen.StyleLibraryType
- Siemens.Engineering.Hmi.Screen.StyleSheetLibraryType
- Siemens.Engineering.Hmi.Tag.HmiUdtLibraryType
- Siemens.Engineering.SW.Blocks.CodeBlockLibraryType
- Siemens.Engineering.SW.Types.PlcTypeLibraryType

Program code

```
TiaPortal tiaPortal = ...;
Project project = tia.Projects.First();
ProjectLibrary library = project.ProjectLibrary;
//Pre-V14SP1
LibraryType libraryType= library.TypeFolder.Types.Find("VBFunction_1");
//V14SP1
VBScriptLibraryType libraryTypeAsVbScript = libraryType as VBScriptLibraryType;
```

5.6.4 Accessing global library languages

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A library is open
  See Opening libraries (Page 205)

Application

You can use language setting navigator to access and manage the Global Library languages.

TIA Portal Openness supports the following access to the global library languages:

- Iterating through supported languages.
- Searching through the collection of supported languages by `System.Globalization.CultureInfo`.
- Accessing individual languages. Each Language object will contain a single read-only attribute Culture of type `System.Globalization.CultureInfo`.
- Accessing a collection of active languages.
- Searching through the collection of active languages by `System.Globalization.CultureInfo`.
- Adding a language to a collection of active languages.
- Removing a language from a collection of active languages.
- Setting an editing language.
- Setting a reference language.

Attribute of global library languages

Global library languages provides the following attribute:

Attribute name	Data type	Writeable	Description
LanguageSettings	Siemens.Engineering.LanguageSettings	r/o	Handles global library languages

Program code: Enumerating global library language

Modify the following program code to enumerate the global library language:

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageSettings languageSettings = globalLibrary.LanguageSettings;
```

Program code: Accessing language setting

Modify the following program code to access the language setting on global library:

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageComposition languages = globalLibrary.LanguageSettings.Languages;
foreach (Language language in languages)
{
// Work with this language
}
```

Program code: Setting global library languages

Modify the following program code to set global library languages. If you set a new supported language through TIA Portal Openness, the language will be added to the active languages collection.

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageSettings languageSettings = globalLibrary.LanguageSettings;
LanguageComposition supportedLanguages = languageSettings.Languages;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
Language supportedGermanLanguage = supportedLanguages.Find(CultureInfo.GetCultureInfo("de-
DE"));
activeLanguages.Add(supportedGermanLanguage);
languageSettings.EditingLanguage = supportedGermanLanguage;
languageSettings.ReferenceLanguage = supportedGermanLanguage;
```

Note

Adding and modifying languages in global library language setting does not modify languages of released versions in the global library. This behaviour holds true for updating global library languages through UI (language editor) or LanguageSettings navigator.

5.6.5 Opening libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application. This requirement is only for accessing project libraries.
  See Opening a project (Page 128)

Application

A global library can be opened using System.IO.FileInfo with a path to the library file on a local storage medium or a network storage. Only user global libraries ca be opened by path. A path obtained from a system global library or a corporate global library can not be used to open it.

As of V14 SP1 global libraries can be opened using the GlobalLibraryInfo. The OpenMode is specified in the GlobalLibraryInfo.

A user global library from a previous version of TIA Portal can be upgraded and opened with the current version of TIA Portal. A global library from V13 or a previous version cannot be opened with upgrade. These libraries have to be upgraded to V13 SP1 first.

Libraries opened using TIA Portal Openness will also be added to the global library collection in the TIA Portal, and will be visible in the user interface of TIA Portal.

Program code: Opening a library using System.IO.FileInfo

Modify the following program code:

```
TiaPortal tia = ...
FileInfo fileInfo = ....

UserGlobalLibrary userLib = tia.GlobalLibraries.Open(fileInfo, OpenMode.ReadWrite);
```

Program code: Opening a library using GlobalLibraryInfo

Modify the following program code:

```
TiaPortal tia = ...
IList<GlobalLibraryInfo> libraryInfos = tia.GlobalLibraries.GetGlobalLibraryInfos();
GlobalLibraryInfo libInfo = ...; //check for the info you need from the list, e.g.
GlobalLibrary libraryOpenedWithInfo;
if (libInfo.Name == "myLibrary")
libraryOpenedWithInfo = tia.GlobalLibraries.Open(libInfo);
```

Program code: Upgrading a library

Modify the following program code:

```
TiaPortal tia = ...
FileInfo fileInfo = .... //library from previous TIA Portal version

UserGlobalLibrary userLib = tia.GlobalLibraries.OpenWithUpgrade(fileInfo);
```

OpenMode

Value	Description
ReadOnly	Read access to the library.
ReadWrite	Read and write access to the library.

5.6.6 Comparing libraries

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- The project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to compare the libraries with each other. For example, you can use the feature to find the differences between versions before updating the new version into the project.

You can use the Openness to do the detailed comparision of selected library objects. It can compare two library objects

- Type object
- Version object
- Master copies

The Openness API allows you to compare two type objects for example version object as well as master copies and returns the detailed content compare result.

Method, Parameter and Enum

The TIA Portal Openness supports the following method with its signature type to compare the library objects :

Method	Return type
DetailedCompareResult CompareTo(LibraryType targetLibraryObject)	Siemens.Engineering.Library.Types.LibraryType
DetailedCompareResult CompareTo(LibraryTypeVersion targetLibraryObject)	Siemens.Engineering.Library.Types.LibraryTypeVersion
DetailedCompareResult CompareTo(MasterCopy targetLibraryObject)	Siemens.Engineering.Library.MasterCopies.MasterCopy

The LibraryCompareResult supports the following navigator in TIA Portal Openness:

Name	Types
Parent	Siemens.Engineering.IEngineeringObject
RootElement	Siemens.Engineering.Library.Compare.LibraryCompareResultElement

The LibraryCompareResultElement supports the following navigator and attribute name in TIA Portal Openness:

Name	Types
Parent	Siemens.Engineering.Library.Compare.LibraryCompareResultElementComposition
RootElement	Siemens.Engineering.Library.Compare.LibraryCompareResultElement

Attribute name	Type
ComparisionResult	Siemens.Engineering.Library.Compare.LibraryCompareResultState
DetailedInformation	System.String
LeftName	System.String
RightName	System.String
Left	Siemens.Engineering.IEngineeringObject
Right	Siemens.Engineering.IEngineeringObject

The LibraryCompareResultState ENUM supports the following ENUM items:

Enum Items	Description
ContainerContentsDifferent	Container contents has one or more differences
ContainerContentsIdentical	Container content is identical
ObjectsDifferent	Objects are different
LeftMissing	Left object is missing
RightMissing	Right object is missing
ObjectsIdentical	Objects are identical

The DetailedCompareResult supports the following navigator in TIA Portal Openness:

Name	Type
Parent	Siemens.Engineering.IEngineeringObject
RootElement	Siemens.Engineering.Library.Compare.LibraryCompareResultElement

The DetailedCompareResultElement supports the following navigator and attribute name in TIA Portal Openness:

Name	Type
Parents	Siemens.Engineering.IEngineeringObject

Attribute name	Type	Description
Description	System.String	Detail compare property name
LeftValue	System.Object	Detail compare left property value. The LeftValue property supports the following types, which are System.Guid, System.Boolean, System.String, System.DateTime
RightValue	System.Object	Detail compare right property value. The RightValue property supports the following types, which are System.Guid, System.Boolean, System.String, System.DateTime.
DetailCompareStatus	Siemens.Engineering.Library.Compare.DetailCompareStatus	Returns the comparison result of the property values based on the default criteria filter applied. (Original library property of version object is not compared by default)

The DetailCompareStatus supports the following ENUM items:

Enum Items	Value
ObjectsDifferent	Objects are different
ObjectsIdentical	Objects are identical
NotCompared	When filters are applied, objects are not compared

Program code

```
...
GlobalLibrary globalLibrary = ...;
ProjectLibrary projectLibrary = Project.ProjectLibrary;
LibraryCompareResult libraryCompareResults =
globalLibrary.CompareToLibrary(projectLibrary);
...
```

Modify the following program code to compare two type objects:

```
...
GlobalLibrary globalLibrary = ...;
ProjectLibrary projectLibrary = Project.ProjectLibrary;
LibraryType leftTypeObject = ProjectLib.TypeFolder.Types.Find(typeName);
LibraryType rightTypeObject = GlobalLib.TypeFolder.Types.Find(typeName);
DetailedCompareResult detailedCompareResult = leftTypeObject.CompareTo(rightTypeObject);
...
```

Modify the following program code to compare two versions:

```
...
GlobalLibrary globalLibrary = ...;
ProjectLibrary projectLibrary = Project.ProjectLibrary;
LibraryTypeVersion leftVersionObject =
ProjectLib.TypeFolder.Types.Find(typeName).Versions[0];
LibraryTypeVersion rightVersionObject =
GlobalLib.TypeFolder.Types.Find(typeName).Versions[0];
DetailedCompareResult detailedCompareResult =
leftVersionObject.CompareTo(rightVersionObject);
...
```

Modify the following program code to compare two master copy objects:

```
...
GlobalLibrary globalLibrary = ...;
ProjectLibrary projectLibrary = Project.ProjectLibrary;
LibraryTypeVersion leftMasterCopy =
ProjectLib.MasterCopyFolder.MasterCopies.Find(masterCopyName);
LibraryTypeVersion rightMasterCopy =
GlobalLib.MasterCopyFolder.MasterCopies.Find(masterCopyName);
DetailedCompareResult detailedCompareResult = leftMasterCopy.CompareTo(rightMasterCopy);
...
```

5.6.7 Creating type from version object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to create a type from the version object. This API functionality will be available through the CreateType operation within the LibraryTypeComposition.

Method and Parameter

The CreateFrom() is overloading API method that accepts the following the parameters and returns LibraryType as datatype.

Parameter name	Return type	Description
sourceVersion	LibraryTypeVersion	Specifies the source from where the user wants to create a type
typeName	String	Specifies a name to the newly created type, the API will check if the value passed by user is not null and not empty. If the provided value meets these criteria, it will be used as the name for the type.
typeMtdv	String	Applies to the Unified faceplate type with having Minimum Target Device Version(MTDV). The API will ignore the typeMTDV parameter passed by the user if the type is other than Unified Faceplate. You can create a new type of MinimumTargetDeviceVersion from a source type: • If the typeMtdv parameter provided by the user will be assigned as the MTDV for the newly created type. However, this assignment will only take place if the value provided in typeMtdv is a supported MTDV value. • If the typeMtdv parameter provided by the user is null or empty, the MTDV type of the newly created type will be the same as the source type.

Program code: Creating type using LibraryTypeComposition

Modify the following program code to create a new type of Minimum Target Device Version frm a source type:

```
private static void createTypeFromVersionObject(project Project)
{
//...
ProjectLibrary projectLibrary = Project.ProjectLibrary;
LibraryTypeComposition projectLibraryTypes = project.ProjectLibrary.TypeFolder.Types;
LibraryType sourceLibraryType = projectLibraryTypes[nameofthetype];
LibraryTypeVersion sourceLibraryTypeVersion = sourceLibraryType.Version[index];
LibraryType createType = projectLibraryTypes.CreateFrom(sourceLibraryTypeVersion);
//Or
LibraryType createType = projectLibraryTypes.CreateFrom(sourceLibraryTypeVersion,typeName);
//Or
LibraryType createType = projectLibraryTypes.CreateFrom(sourceLibraryTypeVersion,typeMtdv);
//Or
LibraryType createType =
projectLibraryTypes.CreateFrom(sourceLibraryTypeVersion,typeMtdv,typeName);
//...
}
```

Exception

The Siemens.Engineering.EngineeringTargetInvocationException will be thrown to user for the following cases:

- To perform CreateFrom operation using a null LibraryTypeVersion object
- To perform CreateFrom operation using a LibraryTypeVersion object, which is in Global Library
- To perform CreateFrom operation using a write-protected LibraryTypeVersion object
- To perform CreateFrom operation using a LibraryTypeVersion object, which is already In-work or In-test
- To perform CreateFrom operation using a LibraryTypeVersion object, with unsupported MinimumTargetDeviceVersion(MTDV)
- To perform CreateFrom operation on LibraryTypeComposition of a GlobalLibrary

5.6.8 Enumerating open libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)

Application

All opened global libraries in the TIA Portal, regardless if they have been opened via API or via user interface can be enumerated.

Global Libraries from previous versions of TIA Portal will not be enumerated if they are opened with write access.

Program code

Modify the following program code to enumerate open global libraries.

```
TiaPortal tia = ...
foreach (GlobalLibrary globLib in tia.GlobalLibraries)
{
    ////work with the global library
}
```

See also

Opening a project (Page 128)

5.6.9 Saving and closing libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A library is open.
  See Opening libraries (Page 205)

Application

User global libraries can be closed or saved. Any changes made to the global library will not be saved automatically. All unsaved changes will be discarded without prompting by closing a global library.

System global libraries and corporate global library cannot be closed or saved.

To save and close a global library:

- Use the Save ( ) method to save a user global library
- Use the SaveAs ( ) method to save a user global library in a different directory
- Use the Close ( ) method to close a user global library

Feature tokens included for SaveAs ( ) are:

- PublicAPI : Necessary for any Published feature
- DenyIfTransaction : Necessary since SaveAs should not be allowed within a transaction as it cannot be undone.

Program code

Modify the following program code to save a user global library:

```
UserGlobalLibrary userLib = ...
// save changes and close library
userLib.Save();
userLib.Close();
```

Modify the following program code to save a user global library in a different location:

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
GlobalLibraryComposition globalLibraryComposition = portal.GlobalLibraries;
//please adapt the path and the extension alx to the installed version of TIA Portal
FileInfo existingLibraryfileInfo = new
FileInfo(@"D:\GlobalLibraries\MyGlobalLibrary\MyGlobalLibrary.alx");
DirectoryInfo targetDirectoryInfo = new
DirectoryInfo(@"D:\GlobalLibraries\GlobalLibrarySaveAs");
UserGlobalLibrary userGlobalLibary =
globalLibraryComposition.Open(existingLibraryfileInfo, OpenMode.ReadWrite);
userGlobalLibary.SaveAs(targetDirectoryInfo);
```

Note

The persistence of the library is changed after SaveAs operation. Thus, after the SaveAs is performed, the newly saved library is available in libraries card. However, the original library on which SaveAs operation was performed, would not be available. The newly saved library would have the same mode as the original library on which SaveAs was performed. This means if the existing library was opened in ReadOnly mode, then the newly saved library in the target path would also open in the ReadOnly mode. Similar is the case for ReadWrite mode.

Modify the following program code to close a user global library:

```
UserGlobalLibrary userLib = ...
// close and discard changes
userLib.Close();
```

5.6.10 Archiving and retrieving a library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  SeeConnecting to the TIA Portal (Page 82)
- A library is open.
  See Opening libraries (Page 205)
- A library is save
  See Saving and closing libraries (Page 212)

Introduction

You can use the TIA Portal Openness to archive the open and save global libraries prior to any further modification to prevent from unintended result so that you canuse the TIA Portal Openenss to retrieve the archived library. You can also share the archived file across the network easily.

Archiving a library

You can use the TIA Portal Openness API interface to archive a user global library. The API is available on the Siemens.Engineering.UserGlobalLibrary object.

The API definition looks like the following:

```
public void Archive(System.IO.DirectoryInfo targetDirectory, string targetName,
Siemens.Engineering.Library.LibraryArchivationMode
archivationMode)
```

The targetName is the name of the file created for archived or non archived. This file may or may not contain any file extensions. If you do not provide any extension or provide an extension apart from "zalx" or "zal14" etc., then the archived file could not be retrieved from TIA Portal outside the Openness API.
For LibraryArchivationMode value as Compressed and DiscardRestorableDataAndCompressed, the archived file name is same as it is provided by you. For LibraryArchivationMode value as None and DiscardRestorableData, the Library file extension is automatically decided by TIA Portal Project Manager component, based on the current version of TIA Portal.

Note

You must have saved the library before calling Archive API. In case the library contains any unsaved changes, archive will throw an EngineeringTargetInvocationException.

Library Archivation Mode

The LibraryArchivationMode enumeration have four values:

LibraryArchivation-Mode	Description
None	• No special action are taken with the orginial files. Mode is similiar to a "save as" operation. • No compressed zip file is created in this mode. • The difference with SaveAs in this case is that the Archive will not change the persistence location to the new Archived folder whereas SaveAs does that.
DiscardRestorableData	• The file storage stores the library in an internal data file and this file grows whenever a library data modification happens. In the case of DiscardRestorableData mode this data file is reorganized (only latest version of the objects are stored and history is removed from the file) and Intermediate data, the files of the IM directory and tmp directory (see Library Directory structure) are not copied to the archive location. • No compressed zip file is created in this mode.
Compressed	The tmp library folder structure, created by Archiving, is compressed into a zip compatible archive. The tmp folder structure is removed after creation of the zip file.
DiscardRestorableDataAndCompressed	The tmp library folder structure, created by Archiving, discards the restorable data and then compressed into a zip compatible archive. The tmp folder structure is removed after creation of the zip file.

Program code: Archiving a library

Modify the following program code to archive a user global library:

```
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
//Please adapt the path and the extension alx to the installed version of TIA Portal
var libraryFilePath = @"E:\Sample1\Sample1.alx";
var userGlobalLibrary = tiaPortal.GlobalLibraries.Open(new FileInfo(LibraryFilePath),
OpenMode.ReadWrite);
var archivePath = @"E:\Archive";
var archiveFileName = "SampleArchive";
userGlobalLibrary.Archive(new DirectoryInfo(archivePath), archiveFileName,
LibraryArchivationMode.Compressed);
```

Retrieving a library

You can use the TIA Portal Openness API interface to retrieve an archived TIA Portal library. You can only retrieve a compressed archive. The API is available on the

Siemens.Engineering.GlobalLibraryComposition object.

The API definition looks like the following:

```
public Siemens.Engineering.Library.UserGlobalLibrary Retrieve(System.IO.FileInfo
sourcePath, System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.OpenMode openMode)
```

Note

You cannot retrieve an archived library with 'LibraryArchivationMode.None' or 'LibraryArchivationMode.DiscardRestorableData' enumeration value.

You can call RetrieveWithUpgrade API for the archived library of a previous TIA Portal version. The API definition looks like the following:

```
public Siemens.Engineering.Library.UserGlobalLibrary
RetrieveWithUpgrade(System.IO.FileInfo sourcePath, System.IO.DirectoryInfo
targetDirectory, Siemens.Engineering.OpenMode openMode)
```

Open Mode

The OpenMode enumeration have two values:

OpenMode	Description
ReadMode	Read access to the library. Data can be read from the library.
ReadWrite	Write access to the library. Data can be written to the library.

Program code: Retrieving a library

Modify the following program code to access to retrieve a library:

```
//Please adapt the path and the extension zalx to the installed version of TIA Portal
var archivePath = @"E:\Archive\Sample1.zalx";
var retrievedLibraryDirectory = @"E:\RetrievedLibraries";
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
tiaPortal.GlobalLibraries.Retrieve(new FileInfo(archivePath), new
DirectoryInfo(retrievedLibraryDirectory), OpenMode.ReadWrite));
```

See also

Opening libraries (Page 205)

5.6.11 Creating global libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)

Introduction

Global libraries can be created via TIA Portal Openness API by calling the Create method on the GlobalLibraryComposition. A UserGlobalLibrary will be returned

Parameters for creating global libraries

Parameter	Data Type	Type	Description
Author	String	Mandatory	Author of a global library.
Comment	String	Optional	Comment of a global library.
Name	String	Optional	Name of a global library
TargetDirectory	DirectoryInfo	Mandatory	Directory that will contain global library folder.

GlobalLibraryComposition.Create

Modify the following program code:

```
TiaPortal tia= ...;
DirectoryInfo targetDirectory = new DirectoryInfo(@"D:\GlobalLibraries");
UserGlobalLibrary globalLibrary =
tia.GlobalLibraries.Create<UserGlobalLibrary>(targetDirectory, "Library1")
```

According to this example:

- a folder "D:\GlobalLibraries\Library1" will be created
- a global library file "D:\GlobalLibraries\Library1\Library1.alXX" will be created

See also

Opening a project (Page 128)

5.6.12 Accessing folders in a library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200)

Introduction

You can use the TIA Portal Openness API interface to access the system folders for types and master copies in a library. You can access types, type versions, master copies and user-defined folders within the system folder.

You can access a user-defined folder at any time using the `Find` method, for example `libTypeUserFolder.Folders.Find("SomeUserFolder");`.

Program code: Accessing system folders

Modify the following program code to access the system folder for types in a library:

```
public static void AccessTypeSystemFolder(ILibrary library)
{
    LibraryTypeSystemFolder libTypeSystemFolder = library.TypeFolder;
}
```

Modify the following program code to access the system folder for master copies in a library:

```
public static void AccessMasterCopySystemFolder(ILibrary library)
{
    MasterCopySystemFolder libMasterCopySystemFolder = library.MasterCopyFolder;
}
```

Program code: Accessing user-defined folders via Find() method

```
...
LibraryTypeUserFolderComposition userFolderComposition = ...
LibraryTypeUserFolder userFolder = userFolderComposition.Find("Name of user folder");
...
```

Program code: Enumerating user defined folders

Modify the following program code to enumerate user-defined subfolders in a system folder for types:

```
public static void EnumerateUserFoldersInTypeSystemFolder(ILibrary library)
{
    // Enumerating user folders in type system folder:
    LibraryTypeSystemFolder libTypeSystemFolder = library.TypeFolder;
    foreach (LibraryTypeUserFolder libTypeUserFolder in libTypeSystemFolder.Folders)
    {
        //...
    }
}
```

Modify the following program code to enumerate user-defined subfolders in a system folder for master copies:

```
public static void EnumerateUserFoldersInMasterCopySystemFolder(ILibrary library)
{
    // Enumerating user folders in master copy system folder:
    MasterCopySystemFolder libMasterCopySystemFolder = library.MasterCopyFolder;
    foreach (MasterCopyUserFolder libMasterCopyUserFolder in
libMasterCopySystemFolder.Folders)
    {
        //..
    }
}
```

Modify the following program code to enumerate user-defined subfolders in a user-defined folder for types:

```
public static void EnumerateAllUserFolders(LibraryTypeUserFolder libUserFolder)
{
    foreach (LibraryTypeUserFolder libSubUserFolder in libUserFolder.Folders)
    {
       EnumerateAllUserFolders(libSubUserFolder);
    }
}
```

Modify the following program code to enumerate user-defined subfolders in a user-defined folder for master copies:

```
public static void EnumerateAllUserFolders(MasterCopyUserFolder libUserFolder)
{
    foreach (MasterCopyUserFolder libSubUserFolder in libUserFolder.Folders)
    {
       EnumerateAllUserFolders(libSubUserFolder);
    }
}
```

Program code: Creating user-defined folders

Modify the following program code to create a user-defined folder for types:

```
var typeFolderComposition = ProjectLibrary.TypeFolder.Folders;
var newTypeUserFolder = typeFolderComposition.Create("NewTypeUserFolder");
```

Modify the following program code to create a user-defined folder for master copies:

```
var masterCopyFolderComposition = projectProjectLibrary.MasterCopyFolder.Folders;
MasterCopyUserFolder newMasterCopyUserFolder =
masterCopyFolderComposition.Create("NewMasterCopyUserFolder);
```

Program code: Renaming user-defined folders

Modify the following program code to create a user-defined folder for types:

```
var typeUserFolder =
project.ProjectLibrary.TypeFolder.Folders.Find("SampleTypeUserFolderName");
typeUserFolder.Name = "NewTypeUserFolderName";
```

```
var typeUserFolder = ProjectLibrary.TypeFolder.Folders.Find("SampleTypeUserFolderName");
typeUserFolder.SetAttributes(new[] {new KeyValuePair<string,object>("Name",
"NewTypeUserFolderName")});
```

Modify the following program code to create a user-defined folder for master copies:

```
var masterCopyUserFolder =
project.ProjectLibrary.MasterCopyFolder.Folders.Find("SampleMasterCopyUserFolderName");
masterCopyUserFolder.Name = "NewMasterCopyUserFolderName";
```

```
var masterCopyUserFolder =
ProjectLibrary.MasterCopyFolder.Folders.Find("SampleMasterCopyUserFolderName");
masterCopyUserFolder.SetAttributes(new[] {new KeyValuePair<string,object>("Name",
"NewMasterCopyUserFolderName")});
```

See also

Accessing master copies (Page 231)

5.6.13 Accessing types

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200).
- You have access to a group for types.
  See Accessing folders in a library (Page 217).

Application

You can access the types contained in a library via the TIA Portal Openness API interface.

- You can enumerate the types.
- You can rename types.
- You can access the following attributes for every type:

Attribute	Data type	Description
Author	String	Returns the name of the author.
Comment	MultilingualText	Returns the comment.
Guid	Guid	Returns the GUID of the type.[1]
Name	String	Returns the name of the type. [2]

[1] You can find an individual type in a library using this attribute. The search is recursive.

[2] You can find an individual type in a folder using this attribute. Subfolders are not included in the search. A type name is not unique. There can be several types with the same name in different groups. However, the type Guid is unique.

Subclasses for library type objects

With TIA Portal Openness API you can access library type objects via sub-classes. The following subclasses are existing:

- Siemens.Engineering.Hmi.Faceplate.FaceplateLibraryType
- Siemens.Engineering.Hmi.Faceplate.FaceplateLibraryTypeVersion
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptLibraryTypeVersion
- Siemens.Engineering.Hmi.RuntimeScripting.CScriptLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.CScriptLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.ScreenLibraryType
- Siemens.Engineering.Hmi.Screen.ScreenLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.StyleLibraryType
- Siemens.Engineering.Hmi.Screen.StyleLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.StyleSheetLibraryTypeVersion
- Siemens.Engineering.Hmi.Tag.HmiUdtLibraryType
- Siemens.Engineering.Hmi.Tag.HmiUdtLibraryTypeVersion
- Siemens.Engineering.SW.Blocks.CodeBlockLibraryType
- Siemens.Engineering.SW.Blocks.CodeBlockLibraryTypeVersion
- Siemens.Engineering.SW.Types.PlcTypeLibraryType
- Siemens.Engineering.SW.Types.PlcTypeLibraryTypeVersion

The following code is an example of how to use the library type sub-classes

```
ProjectLibrary library = project.ProjectLibrary;
VBScriptLibraryType vbScriptType = ...;
VBScriptLibraryType libraryTypeAsVbScript = libraryType as VBScriptLibraryType;
VBScriptLibraryTypeVersion libraryTypeVersionAsVbScript = libraryTypeVersion as
VBScriptLibraryTypeVersion
```

Program code

Modify the following program code to enumerate all types in the system folder of a library:

```
public static void EnumerateTypesInTypesSystemFolder(LibraryTypeSystemFolder
libraryTypeSystemFolder)
{
    foreach (LibraryType libraryType in libraryTypeSystemFolder.Types)
    {
        //...
    }
}
```

Modify the following program code to enumerate all types in a user-defined folder of a library:

```
public static void EnumerateTypesInTypesUserFolder (LibraryTypeUserFolder
libraryTypeUserGroup)
{
    foreach (LibraryType libraryType in libraryTypeUserGroup.Types)
    {
        //...
    }
}
```

Modify the following program code to access the attributes of a type:

```
public static void InspectPropertiesOfType (LibraryType libTypeObject)
{
    string typeAuthor = libTypeObject.Author;
    MultilingualText typeComment = libTypeObject.Comment;
    string typeName = libTypeObject.Name;
    Guid typeGUID = libTypeObject.Guid;
}
```

Modify the following program code to find an individual type by its name or GUID:

```
public static void FindTypeObjectInLibrary(ILibrary library)
{
    // Find type object by its GUID in a given library:
    System.Guid targetGuid = ...;
    LibraryType libTypeByGUID = library.FindType(targetGuid);
    // Find type object by its name in a given group:
    LibraryTypeFolder libTypeSystemFolder = library.TypeFolder;
    LibraryType libTypeByName = libTypeSystemFolder.Types.Find("myTypeObject");
}
```

Modify the following program code to rename a type:

```
// Setting the name attribute
var type = project.ProjectLibrary.TypeFolder.Types.Find("SampleTypeName");
type.Name = "NewTypeName";

//Setting the name attribute dynamically
var type = project.ProjectLibrary.TypeFolder.Types.Find("SampleTypeName");
type.SetAttributes(new[] {new KeyValuePair<string,object>("Name", "NewTypeName")});
```

5.6.14 Accessing type versions

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200).
- You have access to a group for types.
  See Accessing folders in a library (Page 217).

Introduction

You can use the TIA Portal Openness to perform the following functionalities:

- Access the type versions
- Enumerate the type versions of a type
- Determine the type to which a type version belongs
- Enumerate the instances of a type version.
- Create a new instance of a type version.
- Navigate from an instance to its connected version object.

You can access the following attributes for every type version:

Attribute	Data type	Description
Author	String	Returns the name of the author.
Comment	MultilingualText	Returns the comment.
Guid	Guid	Returns the GUID of the type version.[1]
ModifiedDate	DateTime	Returns the date and time at which the type version was set to the "`Committed`" status.
State	LibraryTypeVersionState	Returns the status of the version: • `InWork`: Corresponds to the status "In progress" or "In testing" depending on the associated type. • `Committed`: Corresponds to the status "Released".
TypeObject	LibraryType	Returns the type to which this type version belongs.
VersionNumber	Version	Returns the version number as a three digit version identification, for example, "1.0.0".[2]

[1] You can find an individual type version in a library using this attribute.

[2] You can find an individual type version in a "LibraryTypeVersion" composition using this attribute.

Enumerate all type versions of a type

Modify the following program code:

```
//Enumerate the type versions of a type
public static void EnumerateVersionsInType(LibraryType libraryType)
{
    foreach (LibraryTypeVersion libraryTypeVersion in libraryType.Versions)
    {
        //...
    }
}
```

Accessing the attributes of a type version

Modify the following program code:

```
//Acessing the attributes of a type version
public static void InspectPropertiesOfVersion(LibraryTypeVersion libTypeVersion)
{
    string versionAuthor = libTypeVersion.Author;
    MultilingualText versionComment = libTypeVersion.Comment;
    Guid versionGUID = libTypeVersion.Guid; DateTime versionModifiedDate =
libTypeVersion.ModifiedDate;
    LibraryTypeVersionState versionStateLibrary = libTypeVersion.State;
    LibraryType versionParentObject = libTypeVersion.TypeObject;
    Version versionNumber = libTypeVersion.VersionNumber;
}
```

Creating an instance of a type version

You can create a new instance of a type version. The following objects are supported:

- Blocks (FB/FC)
- PLC user data types
- Screens
- VB scripts

An instance can be created of a type version from global library and project library. When you create the instance of a type version from a global library, the type version is first synchronized with the project library.

A recoverable Exception will be thrown if an instance cannot be created in the target, then . possible reasons are:

- The library type version is in-work
- An instance of the library type version already exists in the target device

Modify the following program code:

```
VBScriptLibraryTypeVersion scriptVersion = ...;
VBScriptComposition vbscripts = ...;

//Using the CreateFrom method to create an instance of the version in the VBScripts
composition
VBScript newScript = vbscripts.CreateFrom(scriptVersion);
```

Modify the following program code:

```
ScreenLibraryTypeVersion screenVersion = ...;
ScreenComposition screens = ...;

//Using the CreateFrom method to create an instance of the version in the screens
composition
Screen newScreen = screens.CreateFrom(screenVersion);
```

Modify the following program code:

```
CodeBlockLibraryTypeVersion blockVersion = ...;
PlcBlockComposition blocks = ...;

//Using the CreateFrom method to create an instance of the version in the blocks composition
PlcBlock newBlock = blocks.CreateFrom(blockVersion);
```

Modify the following program code:

```
PlcTypeLibraryTypeVersion plcTypVersione=...;
PlcTypeComposition types=...;

//Using the CreateFrom method to create an instance of the version in the types composition
PlcType newType = types.CreateFrom(plcTypeVersion);
```

Determining uses of a type version

The following uses are distinguished for type versions:

- The type version uses other type versions from the library.
  Example: A user data type is used in a program block. The program block must have access to the user data type. This means the program block depends on the user data type.
  When you access the Dependencies attribute of CodeBlockLibraryVersion through `GetDependencies()` method, a list of LibraryTypeVersions are returned.
- The type is being used by another type version in the library.
  Example: A user data type is used in a program block. The program block must have access to the user data type. The user data type has the associated program block. The program block depends on the user data type.
  When you access the Dependents attribute of PlcTypeLibraryTypeVersion through `GetDependents()` method, a list of LibraryTypeVersions are returned.

Both attributes return a list that contains objects of the `LibraryTypeVersion` type. If there are no uses, an empty list is returned.

Note

If you use these attributes on type versions with the "`InWork`" status, an exception can be thrown.

Modify the following program code:

```
//Determine the uses of a type version in a library
public static void GetDependenciesAndDependentsOfAVersion(LibraryTypeVersion
libTypeVersion)
{
    IList<LibraryTypeVersion> versionDependents = libTypeVersion.Dependents();
    IList<LibraryTypeVersion> versionDependencies = libTypeVersion.Dependencies();
}
```

Program code

Modify the following program code to determine the type to which a type version belongs:

```
public static void GetParentTypeOfVersion(LibraryTypeVersion libTypeVersion)
{
    LibraryType parentType = libTypeVersion.TypeObject;
}
```

Modify the following program code to determine the master copies that contain instances of a type version:

```
public static void GetMasterCopiesContainingInstances(LibraryTypeVersion libTypeVersion)
{
    MasterCopyAssociation masterCopies = libTypeVersion.MasterCopiesContainingInstances;
}
```

Modify the following program code to find an individual type version by its version number:

```
public static void FindVersionInLibrary(ILibrary library, Guid versionGUID)
{
    LibraryTypeVersion libTypeVersionByVersionNumber = library.FindVersion(versionGUID);
}
```

5.6.15 Accessing blocks located in library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- Opening a project
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to get version information from a library object without instantiating and compiling the object.

The following new Export Action will be provided on the LibraryTypeVersion engineering object to export the version content.

The action will create the Exported xml file at given exportFileInfo.

```
void Export('''FileInfo''' exportFileInfo, '''ExportOptions''' exportOptions)
```

Exception will occur:

- In case of the user exceptions stating "The version data cannot be exported as it in in-work state" and the version to be exported is in "in test" state
- In case of the General user DataExchange for example "FileAlreadyExists" exception stating "The export cannot be made because the file 'D:*.xml' already exists".

Library Type Version enginnering Object

The following additions are made to the Library TypeVersion engineering Object:

1. Export action is provided on LibraryTypeVersion
2. Navigator to navigate to the version content browsable:
   - Navigator name : ContentObject
   - ReadPublicationLevel : System
   - Relation name: Engineering.Library.DefaultVersionContentObject
   - Base navigator will provide a default implementation to provide empty browsable collection as version content. Client can override this navigator and provide versioncontent.
3. "LibraryTypeName" and "LibraryTypeGuid" on LibraryTypeVersion object

Attributes	ReadPublicationLevel
LibraryTypeName	System
LibraryTypeGuid	System

Note

This is necessary because without this info you would not be able to understand that exported version info (in the exported xml file) belongs to which type.

Exported Content

Following content are available in the exported file:

- Library Version Info exported

Attributes	SimaticMLAAccess
Author	ReadWrite
Guid	ReadOnly
Modified Data	ReadOnly
VersionNumber	ReadWrite
LibraryTypeName	ReadOnly
Library TypeGuid	ReadOnly

Navigators Exported:

- Comment

Note

Attributes with SimaticMLAccess readonly get exported only if you choose the exportoption as "ExportOptions.WithReadOnly".

Program code

Modify the following program code to export version content information:

```
Guid g = new Guid("35ad9996-d6d7-40c9-989f-0f0c7e21a7b2");
//Block1
var version = m_Project.ProjectLibrary.FindType(g).Versions[0];
version.Export(new FileInfo(@"D:\ExportCodeBlock.xml"),ExportOptions.WithReadOnly);
```

See also

Opening a project (Page 128)

5.6.16 Accessing instances

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200).
- You have access to a group for types.
  See Accessing folders in a library (Page 217).

Application

You can access instances of type versions via the TIA Portal Openness API interface.

Use the `FindInstances(IInstanceSearchScope searchScope)` method to find all instances of a type version.

You can use the `searchScope` parameter to specify the area of the project to be searched. The following classes implement the `IInstanceSearchScope` interface and can be used to search for instances:

- `PlcSoftware`
- `HmiTarget`

The method returns a list that contains objects of the `LibraryTypeInstanceInfo` type. If there are no instances, an empty list is returned.

Note

Instances of faceplates and HMI user data types are always linked to the associated type version.

Instances of all other objects, such as program blocks or screens, can be linked to a type version.

Enumerate the instances of a type version

Modify the following program code:

```
//Enumerate  the instances of a type version in the project
LibraryTypeVersion version = ...;
PlcSoftware plcSoftware = ...;

IInstanceSearchScope searchScope = plcSoftware as IInstanceSearchScope;

if(searchScope==null)
{
    //No search possible
}

IList<LibraryTypeInstanceInfo> instanceInfos = version.FindInstances(searchScope);
IEnumerable<IEngineeringObject> instances = instanceInfos.Select(instanceInfo =>
instanceInfo.LibraryTypeInstance);
```

Navigate from an instance to its connected version object

Use the LibraryTypeVersion attribute of LibraryTypeInstanceInfo service to navigate from an instance to its connected version object.

The following objects provide the LibraryTypeInstanceInfo service:

- Blocks FB
- Blocks FC
- PLC user data types
- Screens
- VB scripts

If an instance object is not connected to a version object, then it will not provide the "LibraryTypeInstanceInfo" service.

```
FC fc = ...;
//Using LibraryTypeInstanceInfo service

LibraryTypeInstanceInfo instanceInfo = fc.GetService<LibraryTypeInstanceInfo>();
if(instanceInfo != null)
{
    LibraryTypeVersion connectedVersion = instanceInfo.LibraryTypeVersion;
    FC parentFc = instanceInfo.LibraryTypeInstance as FC; //parentFc == fc
}
```

5.6.17 Accessing master copies

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200)
- You have access to a group for master copies.
  See Accessing folders in a library (Page 217)

Introduction

The TIA Portal Openness API interface supports access to master copies in a global library and the project library:

- Creating master copies
- Enumerating master copies in system folders and user-defined folders
- Renaming master copies
- Querying information from master copies
- Querying information from objects in a master copy

Attribute	Data type	Description
Author	String	Returns the name of the author.
ContentDescriptions	MasterCopyContentDescrip-tionComposition	Returns a description for the content of the MasterCopy.
CreationDate	DateTime	Returns the creation date.
Name	String	Returns the name of the master copy.

Program code

Modify the following program code to enumerate all master copies in the system folder of a library:

```
public static void EnumerateMasterCopiesInSystemFolder (MasterCopySystemFolder
masterCopySystemFolder)
{
    foreach (MasterCopy masterCopy in masterCopySystemFolder.MasterCopies)
    {
        //...
    }
}
```

Modify the following program code to access an individual mastercopy by using the find method:

```
...
    MasterCopySystemFolder systemFolder = projectLibrary.MasterCopyFolder;
    MasterCopyComposition mastercopies = systemFolder.MasterCopies;
    MasterCopy masterCopy = mastercopies.Find("Copy of ...");
...
```

Modify the following program code to enumerate master copy groups and subgroups:

```
private static void EnumerateFolder(MasterCopyFolder folder)
{
    EnumerateMasterCopies(folder.MasterCopies);
    foreach (MasterCopyUserFolder subFolder in folder.Folders)
    {
        EnumerateFolder(subFolder); // recursion
    }
}
private static void EnumerateMasterCopies(MasterCopyComposition masterCopies)
{
    foreach (MasterCopy masterCopy in masterCopies)
    {
        ...
    }
}
```

Modify the following program code to access a MasterCopyUserFolder by using the find method:

```
...
    MasterCopyUserFolderComposition userFolderComposition = ...
    MasterCopyUserFolder userFolder = userFolderComposition.Find("Name of user folder");
...
```

Modify the following program code to rename a master copy:

```
//Setting the name attribute
var masterCopy = projectLibrary.MasterCopyFolder.MasterCopies.Find("SampleMasterCopyName");
masterCopy.Name = "NewMasterCopyName";

//Setting the name attribute dynamically
var masterCopy = projectLibrary.MasterCopyFolder.MasterCopies.Find("SampleMasterCopyName");
masterCopy.SetAttributes(new[] {new KeyValuePair<string,object>("Name",
"NewMasterCopyName")});
```

Querying information from master copies

Modify the following program code to get information of a master copy:

```
public static void GetMasterCopyInformation(MasterCopy masterCopy)
{
    string author = masterCopy.Author;
    DateTime creationDate = masterCopy.CreationDate;
    string name = masterCopy.Name;
}
```

Querying information from objects in a master copy

The MasterCopy object contains a navigator called ContentDescriptions, which is a composition of MasterCopyContentDescriptions.

A master copy may contain multiple objects. The MasterCopy object contains ContentDescriptions for each object directly contained in the MasterCopy. If the master copy contains a folder which also contains some items, the MasterCopy object only contains one ContentDescription of the folder.

```
MasterCopy multiObjectMasterCopy = ...;

//Using ContentDescriptions
MasterCopyContentDescriptionComposition masterCopyContentDescriptions =
multiObjectMasterCopy.ContentDescriptions;
MasterCopyContentDescription contentDescription= masterCopyContentDescriptions.First();

string name = contentDescription.ContentName;
Type type = contentDescription.ContentType;
```

5.6.18 Create master copy from a project in library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

If the library is a read-write library, you can create a MasterCopy of an IMasterCopySource at target location.

```
MasterCopy
MasterCopyComposition.Create(Siemens.Engineering.Library.MasterCopies.IMasterCopySource
sourceObject);
```

An EngineeringException will be thrown if:

- The target location is read-only
- Creation of the MasterCopy from the source is rejected by the system

The following items are defined as IMasterCopySources:

- Device - HW
- DeviceItem - HW
- DeviceUserGroup - HW
- CodeBlock - SW
- DataBlock - SW
- PlcBlockUserGroup - SW
- PlcTag - SW
- PlcTagTable - SW
- PlcTagTableUserGroup - SW
- PlcType - SW
- PlcTypeUserGroup - SW
- PlcUnit - SW
- VBScript - HMI
- VBScriptUserFolder - HMI
- Screen - HMI
- ScreenTemplate - HMI
- ScreenTemplateUserFolder - HMI

- ScreenUserFolder - HMI
- Tag - HMI
- TagTable - HMI
- TagUserFolder - HMI

Program code

Use the following program code:

```
// create a master copy from a code block in the project library
public static void Create(Project project, PlcSoftware plcSoftware)
{
    MasterCopySystemFolder masterCopyFolder = project.ProjectLibrary.MasterCopyFolder;
   CodeBlock block = plcSoftware.BlockGroup.Groups[0].Blocks.Find("Block_1") as CodeBlock;
    MasterCopy masterCopy = masterCopyFolder.MasterCopies.Create(block);
}
```

5.6.19 Create an object from a master copy

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Introduction

The TIA Portal Openness API interface supports the use of master copies in the project. You can create an object in the object's composition from a master copy in a project library or a global library using the CreateFrom method.

The return type will correspond to the respective composition's return type.

The CreateFrom method only supports master copies containing single objects. If the composition where the action is called and the source master copy are incompatible (e.g. source master copy contains a plc tag table and the composition is a plc block composition), a recoverable exception will be thrown.

The following compositions are supported:

- Siemens.Engineering.HW.DeviceComposition
- Siemens.Engineering.HW.DeviceItemComposition
- Siemens.Engineering.SW.Blocks.PlcBlockComposition
- Siemens.Engineering.SW.Tags.PlcTagTableComposition

- Siemens.Engineering.SW.Tags.PlcTagComposition
- Siemens.Engineering.SW.Types.PlcTypeComposition
- Siemens.Engineering.SW.TechnologicalObjects.TechnologicalInstanceDBComposition
- Siemens.Engineering.SW.Tags.PlcUserConstantComposition
- Siemens.Engineering.Hmi.Tag.TagTableComposition
- Siemens.Engineering.Hmi.Tag.TagComposition
- Siemens.Engineering.Hmi.Screen.ScreenComposition
- Siemens.Engineering.Hmi.Screen.ScreenTemplateComposition
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptComposition
- Siemens.Engineering.HW.SubnetComposition
- Siemens.Engineering.HW.DeviceUserGroupComposition
- Siemens.Engineering.SW.Blocks.PlcBlockUserGroupComposition
- Siemens.Engineering.SW.ExternalSources.PlcExternalSourceUserGroupComposition
- Siemens.Engineering.SW.Tags.PlcTagTableUserGroupComposition
- Siemens.Engineering.SW.Types.PlcTypeUserGroupComposition

Program code: Create a PLC block from a mastercopy

Modify the following program code to create an PLC block from a master copy in a library:

```
var plcSoftware = ...;
MasterCopy copyOfPlcBlock = ...;
PlcBlock plcSoftware.BlockGroup.Blocks.CreateFrom(copyOfPlcBlock);
```

Program code: Create a device from a mastercopy

Modify the following program code to create a device from a master copy in a library:

```
Project project = ...;
MasterCopy copyOfDevice = ...;
Device newDevice = project.Devices.CreateFrom(copyOfDevice);
```

Program code: Create a device item from a mastercopy

Modify the following program code to create a device item from a master copy in a library:

```
Device device = ...;
MasterCopy copyOfDeviceItem = ...;
DeviceItem newDeviceItem = device.DeviceItems.CreateFrom(copyOfDeviceItem);
```

Program code: Create a subnet from a mastercopy

Modify the following program code to create a subnet from a master copy in a library:

```
Project project = ...;
MasterCopy copyOfSubnet = ...;
Subnet newSubnet = project.Subnets.CreateFrom(copyOfSubnet);
```

Program code: Create a device folder from a mastercopy

Modify the following program code to create a device folder from a master copy in a library:

```
Project project = ...;
MasterCopy copyOfDeviceGroup = ...;
DeviceGroup newDeviceGroup= project.DeviceGroups.CreateFrom(copyOfDeviceGroup);
```

See also

Accessing master copies (Page 231)

5.6.20 Copying master copies

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

The TIA Portal Openness API interface supports copying of master copies within a library and between libraries using the CreateFrom action.

The action will create a new object based on the source master copy and place it in the composition where the action was called. The action will try to create the new master copy with the same name as the source master copy.

If such name is not available, the system will give the new master copy a new name. It will then return the new master copy.

If the composition where the "CreateFrom" action is called is in a read-only global library, a recoverable exception will be thrown.

Program code

Modify the following program code:

```
Project project = ...;
ProjectLibrary projectLibrary = project.ProjectLibrary;
MasterCopy copiedMasterCopy =
projectLibrary.MasterCopyFolder.MasterCopies.CreateFrom(sampleMasterCopy);
```

See also

Accessing master copies (Page 231)

5.6.21 Determining out-of-date type instances

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200)
- You have access to a folder for types.
  See Accessing folders in a library (Page 217).

Application

The TIA Portal Openness API interface allows you to determine type versions which belong to the instances in the open project. The TIA Portal Openness API returns one of the following two states per instance:

- The instance refers to an out-of-date type version.
- The instance refers to the latest type version.

The following rules apply when determining the version:

- You determine the version based on a library and the project that you want to open via the TIA Portal Openness API interface.
- Instances are not updated when you determine the version.

Signature

Use the `UpdateCheck` method to determine the instances of a type version:
`UpdateCheck(Project project, UpdateCheckMode updateCheckMode)`

Parameter	Function
`Project`	Specifies the project in which the type versions of instances are determined.
`UpdateCheckMode`	Specifies the versions that are determined: • `ReportOutOfDateOnly`: Returns only status of the "out of date" type. • `ReportOutOfDateAndUpToDate`: Returns status of the type "out of date" and "up to date".

Result

The devices of the project are scanned from top to bottom when determining the version. Each device is checked to determine whether its configuration data contain an instance of a type version from the specified library. The `UpdateCheck` method returns the result of the version check in hierarchical order.

The table below shows a result of a version check with the parameter `UpdateCheck.ReportOutOfDateAndUpToDate`:

Update check for: HMI_1				
	Update check for library element Screen_1 0.0.3			
		Out-of-date		
			\HMI_1\Screens	Screen_4 0.0.1
			\HMI_1\Screens	Screen_2 0.0.2
		Up-to-date		
			\HMI_1\Screens	Screen_1 0.0.3
			\HMI_1\Screens	Screen_10 0.0.3
Update check for: HMI_2				
	Update check of library element Screen_4 0.0.3			
		Out-of-date		
			\Screens folder1	Screen_02 0.0.1
			\Screens folder1	Screen_07 0.0.2
		Up-to-date		
			\Screens folder1	Screen_05 0.0.3
			\Screens folder1	Screen_08 0.0.3

Program code

There is no difference between project and global libraries in handling the update check.

Modify the following program code to determine the type versions from a global or project library for instances in the project:

```
public static void UpdateCheckOfGlobalLibrary(Project project, ILibrary library)
{
    // check for out of date instances and report only out of date instances in the returned
feedback
    UpdateCheckResult result = library.UpdateCheck(project,
UpdateCheckMode.ReportOutOfDateOnly);

    //Alternatively, check for out of date instances and report both out of date and up to
date instances in the returned feedback
    UpdateCheckResult alternateResult = library.UpdateCheck(project,
UpdateCheckMode.ReportOutOfDateAndUpToDate);

    //Show result
    RecursivelyWriteMessages(result.Messages);

    // Alternatively, show result and access single message parts
    RecursivelyWriteMessageParts(result.Messages);
}
```

Modify the following program code to output the result of the version check and process the messages individually:

```
private static void RecursivelyWriteMessages (UpdateCheckResultMessageComposition
messages, string indent = "")
{
    indent += "\t";
    foreach (UpdateCheckResultMessage message in messages)
    {
        Console.WriteLine(indent + message.Description);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

Modify the following program code to access individual message parts in the result of the version check:

```
private static void RecursivelyWriteMessageParts (UpdateCheckResultMessageComposition
messages, string indent= "")
{
    indent += "\t";
    foreach (UpdateCheckResultMessage message in messages)
    {
        Console.WriteLine(indent + "Full description: " + message.Description);
        foreach (KeyValuePair<string, string> messagePart in message.MessageParts)
        {
             // first level
             // part 1: device name
             // second level:
             // part 1: Name of the type in the global library
             // part 2: version of the type in the global library
             // third level:
             // part 1: title (either "Out-of-date" or "Up-to-date");
             // fourth level:
             // part 1: Path hierarchy to instance
             // part 2: Instance name in project
             // part 3: Version of the instance in the project
             Console.WriteLine(indent + "*Key: {0} Value:{1}", messagePart.Key,
messagePart.Value);
        }
     RecursivelyWriteMessageParts(message.Messages,indent);
    }
}
```

5.6.22 Updating the project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200).
- You have access to a folder for types.
  See Accessing folders in a library (Page 217).

Application

The TIA Portal Openness API interface allows you to update instances of a selected type within a type folder in a project.

When updating, the instances used in the project are updated based on the last released type version. If you start updating the instances from a global library, synchronization is performed beforehand.

Signature

Use the `UpdateProject` method to update instances.

Use the following call for classes which implement the `LibraryTypes` interface:

```
void UpdateProject(IUpdateProjectScope updateProjectScope)
```

Use the following call for classes which implement the `ILibrary` interface:

```
void UpdateProject(IEnumerable<ILibraryTypeOrFolderSelection>
selectedTypesOrFolders, IEnumerable <IUpdateProjectScope>
updateProjectScope)
```

Each call is entered in the log file in the project directory.

Parameter	Function
`IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders`	Specifies the folder or types to be synchronized or their instances in the project to be updated.
`IUpdateProjectScope updateProjectScope` `IEnumerable <IUpdateProjectScope> updateProjectScope`	Specifies the object(s) in the project in which the uses of instances are to be updated. The following objects are supported: • `PlcSoftware` • `HmiTarget`

Program code

Modify the following program code to update instances of selected types within a type folder:

```
private static void UpdateInstances(ILibrary myLibrary, LibraryTypeFolder
singleFolderContainingTypes, LibraryType singleType, PlcSoftware plcSoftware, HmiTarget
hmiTarget)
{
    //Update Instances of multiple types (subset of types and folders)
    IUpdateProjectScope[] updateProjectScopes =
    {
        plcSoftware as IUpdateProjectScope, hmiTarget as IUpdateProjectScope
    };
    myLibrary.UpdateProject(new ILibraryTypeOrFolderSelection[] {singleType,
singleFolderContainingTypes}, updateProjectScopes);
    //Update Instances of multiple types (all types in library)
    myLibrary.UpdateProject(new[] {myLibrary.TypeFolder}, updateProjectScopes);
}
```

5.6.23 Updating a library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200).
- You have access to a folder for types.
  See Accessing folders in a library (Page 217).

Application

The TIA Portal Openness API interface supports the following updates in the project library:

- Synchronize selected types between libraries.

The folder structure is not adapted when you perform synchronization. The types to be updated are identified by their GUID and updated:

- If a type in a library includes a type version that is missing in the library to be updated, the type version is copied.
- If a type in a library includes a type version with the different GUID, the process is aborted and an Exception is thrown.

Signature

Use the `UpdateLibrary` method to synchronize type versions.

Use the following call for classes which implement the `LibraryTypes` interface:

```
void UpdateLibrary(ILibrary targetLibrary)
```

Use the following call for classes which implement the `ILibrary` interface:

```
void UpdateLibrary(IEnumerable<LibraryTypeOrFolderSelection>
selectedTypesOrFolders, ILibrary targetLibrary)
```

Parameter	Function
`IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders`	Specifies the folder or types to be synchronized or their instances in the project to be updated.
`ILibrary targetLibrary`	Specifies the library whose contents will be synchronized with a library. If source library and destination library are identical, an exception is thrown.

Program code

Modify the following program code to synchronize a type from the project library with a global library:

```
sourceType.UpdateLibrary(projectLibrary);
```

Modify the following program code to synchronize selected types within a type folder between a global library and the project library:

```
globalLibrary.UpdateLibrary(new[]{globalLibrary.TypeFolder}, projectLibrary);
```

5.6.24 Cleaning up library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness application to delete the unused versions of the specified types in a library and delete all the types in a specified folder in a library.

The Cleanup() is available for project and global libraries for which libraries must be read/write. This feature is analogus to the Clean up Library feature present in TIA Portal user interface.

The Cleanup() performs function based on the specified CleanupLibraryMode flag.

A version is cleaned up based on the specified behavior of the two CleanupLibraryModes:

CleanupLibraryMode	Description
CleanupLibraryMode.AllowTypeDeletion	Allow deletion of all the versions on a type to be cleaned up. If all versions on a type have been cleaned up, the type too will be cleaned up.
CleanupLibraryMode.PreserveHighestVersion	Preserve the highest version on a type

Program code

Modify the following program code to invoke cleanup action for a subset of types and folders:

```
...
ILibrary myLibrary = ...;
CleanupLibraryMode cleanupLibraryMode = ...;
ILibraryType typeA = ...;//from myLibrary
ILibraryType typeB = ...; //from myLibrary
LibraryTypeFolder singleFolderContainingTypes = ...;//from myLibrary
myLibrary.Cleanup(new[]{typeA, singleFolderContainingTypes}, cleanupLibraryMode);
...
```

Modify the following program code to invoke cleanup action for all types in a library:

```
...
ILibrary myLibrary = ...;
CleanupLibraryMode cleanupLibraryMode = ...;
myLibrary.Cleanup(new [] {myLibrary.TypeFolder}, cleanupLibraryMode);
...
```

5.6.25 Harmonize project from project library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to harmonize the names and paths of type instances with the types in a specified library. The HarmonizeProject() is available for both project and global libraries. This feature is analogous to the Harmonize feature present in TIA Portal user interface.

You can specifiy which devices in the project to harmonize along with the option to harmonize names, paths, or both. If the HarmonizeProject feature is called on a global library, the project library will be first harmonized and then the project.

The Harmonize is handled according to the chosen HarmonizeProjectOptions:

HarmonizeProjectOptions	Description
HarmonizePaths	Harmonize paths of the specified types or the types under the specified folders. The instances in the project are renamed to the type names of the library
HarmonizeProjectOptions.HarmonizeNames	Harmonize names of the specified types or the types under the specified folders. The folder structure of the library types are applied to the project instances

HarmonizeProjectOptions	Description
HarmonizeProjectOptions.HarmonizeNames | HarmonizeProjectOptions.HarmonizePaths	Harmonize names and paths of types selected or the types under a selected folder
HarmonizeProjectOptions.None	This is not to be used and exception is thrown on usage

Introduction

Modify the following program code to perform Harmonize names for a subset of types and folders:

```
...
ILibrary myLibrary = ...;
HarmonizeProjectOptions harmonizeProjectOptions = ...;
ILibraryType typeA = ...; //from myLibrary
ILibraryType typeB = ...; //from myLibrary
LibraryTypeFolder singleFolderContainingTypes = ...; //from myLibrary
IUpdateProjectScope harmonizeProjectScope = ..... //scope of device in project to be
harmonized
myLibrary.HarmonizeProject(new[]{typeA, singleFolderContainingTypes}, new[]
{harmonizeProjectScope}, HarmonizeProjectOptions.HarmonizeNames);
...
```

Modify the following program code to perform Harmonize paths for a subset of types and folders:

```
...
ILibrary myLibrary = ...;
HarmonizeProjectOptions harmonizeProjectOptions = ...;
ILibraryType typeA = ...; //from myLibrary
ILibraryType typeB = ...; //from myLibrary
LibraryTypeFolder singleFolderContainingTypes = ...; //from myLibrary
IUpdateProjectScope harmonizeProjectScope = ..... //scope of device in project to be
harmonized
myLibrary.HarmonizeProject(new[]{typeA, singleFolderContainingTypes}, new[]
{harmonizeProjectScope}, HarmonizeProjectOptions.HarmonizePaths);
...
```

Modify the following program code to perform Harmonize names and paths for a subset of types and folders:\

```
...
ILibrary myLibrary = ...;
HarmonizeProjectOptions harmonizeProjectOptions = ...;
ILibraryType typeA = ...; //from myLibrary
ILibraryType typeB = ...; //from myLibrary
LibraryTypeFolder singleFolderContainingTypes = ...; //from myLibrary
IUpdateProjectScope harmonizeProjectScope = ..... //scope of device in project to be
harmonized
myLibrary.HarmonizeProject(new[]{typeA, singleFolderContainingTypes}, new[]
{harmonizeProjectScope}, HarmonizeProjectOptions.HarmonizeNames |
HarmonizeProjectOptions.HarmonizePaths);
...
```

Exception Handling

The exception scenarios are stated below:

NULL in sourceTypesAndFolder collection:

```
var projectLibrary = Project.ProjectLibrary;
var type1 = projectLibrary.Types.FindType("Type1"); // if this statement yields NULL
projectLibrary.HarmonizeProject(new[]{type1}, new[] {harmonizeProjectScope},
HarmonizeProjectOptions.HarmonizeNames); // throws EngineeringTargetInvocationException
```

NULL sourceTypesAndFolder collection:

```
projectLibrary.HarmonizeProject(null, new[] { harmonizeProjectScope },
HarmonizeProjectOptions.HarmonizeNames); // throws EngineeringTargetInvocationException
```

Empty sourceTypesAndFolder collection

```
projectLibrary.HarmonizeProject(new[]{}, new[] {harmonizeProjectScope},
HarmonizeProjectOptions.HarmonizeNames); // throws EngineeringTargetInvocationException
```

NULL in harmonizeProjectScope collection

```
var projectLibrary = Project.ProjectLibrary;
var type1 = projectLibrary.Types.FindType("Type1");
projectLibrary.HarmonizeProject(new[]{type1}, new[] { null },
HarmonizeProjectOptions.HarmonizeNames); // throws EngineeringTargetInvocationException
```

HarmonizeProjectOptions is None

```
var projectLibrary = Project.ProjectLibrary;
var type1 = projectLibrary.Types.FindType("Type1");// if this statement yields NULL
projectLibrary.HarmonizeProject(new[]{type1}, new[] { harmonizeProjectScope },
HarmonizeProjectOptions.None); // throws EngineeringTargetInvocationException
```

sourceTypesAndFolder from different library

```
var type1 = projectLibrary.Types.FindType("Type1");
var type2 = globalLibrary.Types.FindType("Type2");
projectLibrary.HarmonizeProject(new[]{type1, type2}, new[] { harmonizeProjectScope },
HarmonizeProjectOptions.None); // throws EngineeringTargetInvocationException
```

Other cases of EngineeringTargetInvocationException are as below:

- The project is in ReadOnly state
- The Library is in ReadOnly state
- The harmonizeScope are from different Projects
- The harmonizeScope Collection is in ReadOnly state

5.6.26 Updating Structure during Library Update

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to update or retain the project and global library structure even if there are conflicts. You can also use the TIA Portal Openness to retain or delete unused

versions from target library.

The UpdateLibrary() is defined with the following signature on Project Library and Global Library:

public void UpdateLibrary(IEnumerable<ILibraryTypeOrFolderSelection> sourceTypesAndFolders, ILibrary targetLibrary, ForceUpdateMode forceUpdateMode, DeleteUnusedVersionsMode updateMode, StructureConflictResolutionMode structureConflictResolutionMode);

The UpdateLibrary() is overloaded with another definition on Single Types with the following signature :

public void UpdateLibrary(ILibrary targetLibrary, DeleteUnusedVersionsMode updateMode, StructureConflictResolutionMode structureConflictResolutionMode,ForceUpdateMode forceUpdateMode);

The UpdateLibrary() accepts the following parameter definition to perform different operations:

Parameter	Definition
DeleteUnusedVersionsMode	To control whether to delete/retain unused versions from the Target library. The DeleteUnusedVersionMode performs operations based on the ENUM value provided as input: • DeleteUnusedVersionsMode.AutomaticallyDelete : Unused versions will be deleted in the Target Library if there are unused version in target libraries • DeleteUnusedVersionsMode.DoNotDelete : Unused versions will not be deleted in the Target Library if there are unused version in target libraries
StructureConflictResolutionMode	To control whether to update the target library structure, retain the existing structure or to cancel the operation if there is a structural conflict. The StructureConflictResolutionMode performs operations based on the ENUM value provided as input: • StructureConflictResolutionMode.UpdateStructure : Source Library structure will be updated in the Target Library if there are structural differnces between source and target libraries • StructureConflictResolutionMode.RetainStructure : Source Library structure will be same in the Target Library if there are structural differnces between source and target libraries • StructureConflictResolutionMode.CancelIfStructureConflicts : Throws the EngineeringTargetInvocationException and cancel the operation If a structure conflict occurs
ForceUpdateMode	To control whether to set the updated version on the target library as default/latest. The ForceUpdateMode performs operations based on the ENUM value provided as input: • ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault : Default version in source library is updated to target and the updated version is marked as default version in target only if source version number is higher than target version number • ForceUpdateMode.ForceSetAnyUpdatedVersionAsDefault : Default version in source library is updated to target as default or latest version, irrespective of version number • ForceUpdateMode.NoDefaultVersionChange : This enum value should be used during update from project library to project where force update option is not applicable

Program code: Single Type Update Library

The UpdateLibrary API is part of Siemens.Engineering.Library.Types.LibraryType interface and will accept a "singleselection" of types.

Modify the program code to perform update libraryUpdateLibrary for a single type from Project Library to Global Library(delete unused version , update existing target library structure and force update target version as default irrespective of version number)

```
...
ILibrary myProjectLibrary = Project.ProjectLibrary;
LibraryTypeUserFolder singleFolderContainingTypes =
myProjectLibrary.TypeFolder.Folders.Find("folder1");
ILibraryType typeA = singleFolderContainingTypes.Types.Find("block1");
// block1 has lower version.
DeleteUnusedVersionsMode deleteUnusedVersionsMode =
DeleteUnusedVersionsMode.AutomaticallyDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.UpdateStructure;
ForceUpdateMode forceUpdateMode = ForceUpdateMode.ForceSetAnyUpdatedVersionAsDefault;
ILibrary globalLibrary = ...;
typeA.UpdateLibrary(globalLibrary, deleteUnusedVersionsMode,
structureConflictResolutionMode, forceUpdateMode);
...
```

Modify the following program code to perform UpdateLibrary for a single type from Project Library to Global Library(do not delete unused version, retain existing target library structure and do not perform force update):

```
...
ILibrary globalLibrary = ...;
LibraryTypeUserFolder singleFolderContainingTypes =
globalLibrary.TypeFolder.Folders.Find("folder1");
ILibraryType typeA = singleFolderContainingTypes.Types.Find("block1");// block1 has lower
version.
DeleteUnusedVersionsMode deleteUnusedVersionsMode = DeleteUnusedVersionsMode.DoNotDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.RetainStructure;
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
ILibrary myProjectLibrary = Project.ProjectLibrary;
typeA.UpdateLibrary(myProjectLibrary, deleteUnusedVersionsMode,
structureConflictResolutionMode, forceUpdateMode);
...
```

Modify the following program code to perform UpdateLibrary for a single type from Global Library to Project Library(do not delete unused version, retain existing target library structure and do not perform force update):

```
...
ILibrary globalLibrary = ...;
LibraryTypeUserFolder singleFolderContainingTypes =
globalLibrary.TypeFolder.Folders.Find("folder1");
ILibraryType typeA = singleFolderContainingTypes.Types.Find("block1"); // block1 has lower
version.
DeleteUnusedVersionsMode deleteUnusedVersionsMode = DeleteUnusedVersionsMode.DoNotDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.RetainStructure;
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
ILibrary myProjectLibrary = Project.ProjectLibrary;
typeA.UpdateLibrary(myProjectLibrary, deleteUnusedVersionsMode,
structureConflictResolutionMode, forceUpdateMode);
...
```

Modify the following program code to perform UpdateLibrary for a single type from Global Library to Project Library(delete unused version, update existing target library structure and do not perform force update):

```
...
ILibrary globalLibrary = ...;
LibraryTypeUserFolder singleFolderContainingTypes =
globalLibrary.TypeFolder.Folders.Find("folder1");
ILibraryType typeA = singleFolderContainingTypes.Types.Find("block1"); // block1 has lower
version.
DeleteUnusedVersionsMode deleteUnusedVersionsMode =
DeleteUnusedVersionsMode.AutomaticallyDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.UpdateStructure;
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
ILibrary myProjectLibrary = Project.ProjectLibrary;
typeA.UpdateLibrary(myProjectLibrary, deleteUnusedVersionsMode,
structureConflictResolutionMode);
...
```

Program code: Multiple Types Update Library

The UpdateLibrary API is part of Siemens.Engineering.Library.ILibrary interface and will accept a "multiselection" of types and type folders.

Modify the following program code to perform UpdateLibrary for a single type from Project Library to Global Library:

```
...ILibrary myProjectLibrary = Project.ProjectLibrary;
LibraryTypeUserFolder singleFolderContainingTypes =
myProjectLibrary.TypeFolder.Folders.Find("folder1");
ILibraryType typeA = singleFolderContainingTypes.Types.Find("block1"); // block1 has lower
version.
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
DeleteUnusedVersionsMode deleteUnusedVersionsMode = DeleteUnusedVersionsMode.DoNotDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.UpdateStructure;
ILibrary globalLibrary;
myProjectLibrary.UpdateLibrary(new[] { typeA }, globalLibrary, forceUpdateMode,
deleteUnusedVersionsMode, structureConflictResolutionMode);
...
```

Modify the following program code to perform UpdateLibrary for multiple types from Project Library to Global Library:

```
...
ILibrary myProjectLibrary = Project.ProjectLibrary;
LibraryTypeUserFolder singleFolderContainingTypes =
myProjectLibrary.TypeFolder.Folders.Find("folder1");
ILibraryType typeA = singleFolderContainingTypes.Types.Find("block1"); // block1 has lower
version.
ILibraryType typeB = ...;
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
DeleteUnusedVersionsMode deleteUnusedVersionsMode = DeleteUnusedVersionsMode.DoNotDelete;
StructureConflictResolutionMode structureConflictResolutionMode=
StructureConflictResolutionMode.UpdateStructure;
ILibrary globalLibrary;myProjectLibrary.UpdateLibrary(new[] { typeA, typeB },
globalLibrary, forceUpdateMode, deleteUnusedVersionsMode, structureConflictResolutionMode);
...
```

Modify the following program code to perform UpdateLibrary for a user folder containing type from Project Library to Global Library:

```
...
ILibrary myProjectLibrary = Project.ProjectLibrary;
LibraryTypeUserFolder singleFolderContainingTypes =
myProjectLibrary.TypeFolder.Folders.Find("folder1");
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
DeleteUnusedVersionsMode deleteUnusedVersionsMode = DeleteUnusedVersionsMode.DoNotDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.UpdateStructure;
ILibrary globalLibrary;
myProjectLibrary.UpdateLibrary(new[]
{ (ILibraryTypeOrFolderSelection)singleFolderContainingTypes }, globalLibrary,
forceUpdateMode, deleteUnusedVersionsMode, structureConflictResolutionMode);
...
```

Modify the following program code to perform UpdateLibrary on "Types" system folder from Project Library to Global Library:

```
...
ILibrary myProjectLibrary = Project.ProjectLibrary;
LibraryTypeSystemFolder typeFolder = myProjectLibrary.TypeFolder;
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
DeleteUnusedVersionsMode deleteUnusedVersionsMode = DeleteUnusedVersionsMode.DoNotDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.RetainStructure;
ILibrary globalLibrary;
myProjectLibrary.UpdateLibrary(new[] { (ILibraryTypeOrFolderSelection)typeFolder },
globalLibrary, forceUpdateMode, deleteUnusedVersionsMode, structureConflictResolutionMode);
...
```

Modify the following program code to perform UpdateLibrary for a single type from Global Library to Project Library:

```
...
ILibrary globalLibrary = ...;
LibraryTypeUserFolder singleFolderContainingTypes =
globalLibrary.TypeFolder.Folders.Find("folder1");
ILibraryType typeA = singleFolderContainingTypes.Types.Find("block1"); // block1 has lower
version.
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
DeleteUnusedVersionsMode deleteUnusedVersionsMode = DeleteUnusedVersionsMode.DoNotDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.UpdateStructure;
ILibrary myProjectLibrary = Project.ProjectLibrary;
globalLibrary.UpdateLibrary(new[] { typeA }, myProjectLibrary, forceUpdateMode,
deleteUnusedVersionsMode, structureConflictResolutionMode);
...
```

Modify the following program code to perform UpdateLibrary for a single type from Global Library to Global Library:

```
...
ILibrary globalLibrarySource = ...;
LibraryTypeUserFolder singleFolderContainingTypes =
globalLibrarySource.TypeFolder.Folders.Find("folder1");
ILibraryType typeA = singleFolderContainingTypes.Types.Find("block1"); // block1 has lower
version.
ForceUpdateMode forceUpdateMode = ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault;
DeleteUnusedVersionsMode deleteUnusedVersionsMode = DeleteUnusedVersionsMode.DoNotDelete;
StructureConflictResolutionMode structureConflictResolutionMode =
StructureConflictResolutionMode.UpdateStructure;
ILibrary globalLibraryTarget = ...;
globalLibrarySource.UpdateLibrary(new[] { typeA }, globalLibraryTarget, forceUpdateMode,
deleteUnusedVersionsMode, structureConflictResolutionMode);
...
```

Exception handling for Single Type Exception Handling

Some of the exception scenarios are stated below:

StructureConflictResolutionMode is CancelIfStructureConflicts

```
var myProjectLibrary = Project.ProjectLibrary;
LibraryTypeSystemFolder typeFolder =
myProjectLibrary.TypeFolder.Folders.Find("folder1"); //from myProjectLibrary
var type1 = typeFolder.Types.FindType("Type1"); // if this structure is not available in
Global Library
type1.UpdateLibrary(globalLibrary, DeleteUnusedVersionsMode.DoNotDeleteUnusedVersions,
StructureConflictResolutionMode.CancelIfStructureConflicts,
ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault); // throws
EngineeringTargetInvocationException
```

Target Library is NULL.

```
var myProjectLibrary = Project.ProjectLibrary;
var type1 = projectLibrary.Types.FindType("Type1");
ILibrary globalLibrary = Null;
type1.UpdateLibrary(globalLibrary, DeleteUnusedVersionsMode.DoNotDeleteUnusedVersions,
StructureConflictResolutionMode.RetainStructure,
ForceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault); // throws
EngineeringTargetInvocationException
```

Bad Enum values throws EngineeringTargetInvocationException:

```
var myProjectLibrary = Project.ProjectLibrary;
LibraryTypeSystemFolder typeFolder = myProjectLibrary.TypeFolder.Folders.Find("folder1");//
from myProjectLibrary
var type1 = typeFolder.Types.FindType("Type1"); // if this structure is not available in
Global Library
type1 .UpdateLibrary(globalLibrary, (DeleteUnusedVersionsMode)42,
StructureConflictResolutionMode.RetainStructure, (ForceUpdateMode)22); // throws
EngineeringTargetInvocationException
```

Exception handling for Multiple Type Exception Handling

Some of the exception scenarios are stated below:

NULL in sourceTypesAndFolder collection

```
var myProjectLibrary = Project.ProjectLibrary;
var type1 = projectLibrary.Types.FindType("Type1"); // if this statement yields NULL
myProjectLibrary.UpdateLibrary(new[] { type1 }, globalLibrary,
forceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault,
DeleteUnusedVersionsMode.DoNotDeleteUnusedVersions,
StructureConflictResolutionMode.RetainStructure); // throws
EngineeringTargetInvocationException
```

StructureConflictResolutionMode is CancelIfStructureConflicts

```
var myProjectLibrary = Project.ProjectLibrary;
LibraryTypeSystemFolder typeFolder =
myProjectLibrary.TypeFolder.Folders.Find("folder1"); //from myProjectLibrary
var type1 = typeFolder.Types.FindType("Type1"); // if this structure is not available in
Global Library
myProjectLibrary.UpdateLibrary(new[] { type1 }, globalLibrary,
forceUpdateMode.SetOnlyHigherUpdatedVersionAsDefault,
DeleteUnusedVersionsMode.DoNotDeleteUnusedVersions,
StructureConflictResolutionMode.CancelIfStructureConflicts); // throws
EngineeringTargetInvocationException
```

Bad Enum values throws EngineeringTargetInvocationException

```
var myProjectLibrary = Project.ProjectLibrary;
LibraryTypeSystemFolder typeFolder =
myProjectLibrary.TypeFolder.Folders.Find("folder1"); //from myProjectLibrary
var type1 = typeFolder.Types.FindType("Type1"); // if this structure is not available in
Global Library
myProjectLibrary.UpdateLibrary(new[] { type1 }, globalLibrary, (ForceUpdateMode)22,
(DeleteUnusedVersionsMode)42, StructureConflictResolutionMode.RetainStructure); // throws
EngineeringTargetInvocationException
```

Note

Other cases in which EngineeringTargetInvocationException is thrown are as below:

- Any Update Library operations targeting a read-only global library will result in an exception being thrown.
- An exception will be thrown if any of the selected types or folders are null or not part of the source library.

See also

Opening a project (Page 128)

5.6.27 Updating libraries type

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- The project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to opt out a type from updating to project library or project from global library. The feature helps you to update project or project library with types that need an update and ignore types that are not required.

The feature also helps you to distribute linked libraries from different vendors and update project or project library when the vendor provides additional functionality and is not waiting for entire library to be delivered by all the vendors.

You can use the SetForUpdate attribute only for non write protected global library types where as Get operation can be performed on all the library types. The SetForUpdate always returns True for types in project library.

The following API is used to mark the type which are required to be updated to project library or project.

Attribute name	Service Provider	Description
SetForUpdate	LibraryType	To set or get the distributable property of a Library Type

Note

The Siemens.Engineering.EngineeringNotSupportedException will be thrown to you if you try

- To perform a set operation on SetForUpdate attribute for types in project library
- To perform a set operation on SetForUpdate attribute for types which are write protected

Program code

Modify the following program code to mark the library type for update using SetForUpdate attribute:

```
using Siemens.Engineering;
...
{
     TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
     string globalLibrrayPath =
@"C:\Proj\GlobalLibrary_SetForUpdate\GlobalLibrary_SetForUpdate.al18";
     FileInfo globalLibraryFileInfo = new FileInfo(globalLibrrayPath);
     var gl = portal.GlobalLibraries.Open(globalLibraryFileInfo, OpenMode.ReadWrite);
     var globalLibTypes = gl.TypeFolder.Types;
     //Get 'SetForUpdate' value for a specific library type
     Console.WriteLine(string.Format(":::GloblaLibrary:::SetforUpdateValue::::{0}",
globalLibTypes[0].SetForUpdate.ToString()));
     //Set 'SetForUpdate' value for a specific library type
      globalLibTypes[0].SetForUpdate = true;
}
```

5.6.28 Deleting library content

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)
- You have access to the required library.
  See Accessing global libraries (Page 200).
- You have access to a folder for types.
  See Accessing folders in a library (Page 217).

Introduction

You can delete the following project library content using the TIA Portal Openness API interface:

- Types
- Type version
- User-defined folders for types
- Master copies
- User-defined folders for master copies

Note

Deleting of types and user-defined type folders

If you want to delete a type or user-defined folder type, the "Rules for deleting versions" must be met. You can always delete an empty type folder.

Note

Rules for deleting versions

You can only delete versions with Committed status. The following rules also apply when deleting versions:

- If a new version with the InWork status has just been created from a version with Committed status , you can only delete the version with Committed status when the new version is discarded or it obtains the "Committed" status.
- If a type only has one version, the type is deleted as well.
- If Version A is dependent on Version B of another type, first delete Version A and then Version B.
- If there are instances of Version A, you can only delete Version A if the instances are deleted as well. If an instance is also contained in a master copy, the master copy is deleted as well.

Program code

Modify the following program code to delete types or user-defined type folders:

```
public static void DeleteMultipleTypesOrTypeUserFolders(ILibrary library)
{
    LibraryType t1 = library.TypeFolder.Types.Find("type1");
    LibraryType t2 = library.TypeFolder.Types.Find("type2");
    LibraryTypeUserFolder f1 = library.TypeFolder.Folders.Find("folder1");
    t1.Delete();
    t2.Delete();
    f1.Delete();
}
```

Modify the following program code to delete an individual type or user-defined type folder:

```
public static void DeleteSingleTypeOrTypeUserFolder(ILibrary library)
{
    //Delete a single type
    LibraryType t1 = library.TypeFolder.Types.Find("type1");
    t1.Delete();

    //Delete a single folder
    LibraryTypeFolder parentFolder = library.TypeFolder;
    LibraryTypeUserFolder f1 = parentFolder.Folders.Find("folder1");
    f1.Delete();
}
```

Modify the following program code to delete a version:

```
public static void DeleteVersion(ILibrary library)
{
    LibraryType singleType = library.TypeFolder.Types.Find("type1");
    LibraryTypeVersion version1 = singleType.Versions.Find(new System.Version(1, 0, 0));
    version1.Delete();
}
```

Modify the following program code to delete a master copy or a user-defined master copy folder:

```
public static void DeleteMasterCopies(ILibrary library)
{
    // Delete master copy
    MasterCopy masterCopy = library.MasterCopyFolder.MasterCopies.Find("myMasterCopy");
    masterCopy.Delete();

    // Delete master copy user folder
    MasterCopyUserFolder masterUserFolder =
library.MasterCopyFolder.Folders.Find("myFolder");
    masterUserFolder.Delete();
}
```

See also

Accessing master copies (Page 231)

5.6.29 Setting default version of a type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 82)

Introduction

You can use the TIA Portal Openness to set any released version as default version of that type for project library and user global library.

Default version is considered as preferred version of a type.

Program code

```
ILibrary myLibrary = ....;
LibraryType typeA = ...;// from myLibrary
LibraryTypeVersion versionA = ...;//from typeA
versionA.setAsDefault(); // set the version as a default version of the type.
```

Exception Handling

An exception "EngineeringTargetInvocationException" will be thrown to you if a failure happened during the invocation of SetAsDefault API.

You can encounter exception in the following scenarios:

- In project library, when SetAsDefault() is called on in-test/in-work version
- When Global Library is read-only

See also

Opening a project (Page 128)

5.6.30 Getting default version of a type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to get default version of a type for project library and global library. You can also use the TIA Portal Openness application to check if a version is default or not. Default version is considered as preferred version of a type.

Program code

```
//...
ILibrary myLibrary = ...;
LibraryType typeA = ...// from myLibrary
var typeADefaultVersion = typeA.Versions.First((version) => version.IsDefault);
//from default version , user can get details related to version like Author, Guid, etc.
var nameofType = typeADefaultVersion.LibraryTypeName;
var guid = typeADefaultVersion.Guid;
var versionNumber = typeADefaultVersion.VersionNumber;
var authoroftheVersion = typeADefaultVersion.Author;
var status = typeADefaultVersion.State;
//...
```

See also

Opening a project (Page 128)

5.6.31 Improved types consistency state

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to check types for a inconsistencies. The Types can have different consistency status. The Type is Consistent, if Dependent Type is using default version of dependency. The Type is Inconsistent if Dependent Type is not using the default version of dependency type.

The folder level consistency status shows if any type under the structure is consistent or inconsistent.

The Status API is part of Siemens.Engineering.Library.Types.LibraryType and Siemens.Engineering.Library.Types.LibraryTypeFolder class and will be of ConsistencyStatustype.

To allow TIA Portal Openness user to check the consistency status of type/folders a "Status" attribute will be introduced for the following:

- SystemFolder
- UserFolder
- Type

Using this, you can prepare the list of inconsistent types to perform special operations on them:

```
var listOfInconsistentTypes = Folder.Types.Where(x => x.Status ==
ConsistencyStatus.DefaultVersionInconsistent);
```

ConsistencyStatus enum supports below mentioned values:

Value	Description
DefaultVersionInConsistent	Dependent Types are not using the default version of Dependency Types
Consistent	Dependent Types are using the defautt version of Dependency Types

Program code

Modify the following program code to read status attribute support on "Types" system folder from Project Library:

```
...
ILibrary myProjectLibrary = Project.ProjectLibrary;
LibraryTypeSystemFolder typeFolder = myProjectLibrary.TypeFolder;
var systemTypeFolderConsistencyStatus = typeFolder.Status;
if(systemTypeFolderConsistencyStatus == ConsistencyStatus.DefaultVersionInConsistent)
{
//Do something.
}
// List of InConsistentType
var listOfInCosistentTypes = typeFolder.Types.Where(x => x.Status ==
ConsistencyStatus.InConsistent);
foreach(var type in listOfInCosistentTypes)
{// From type, user can get details like Author, Guid etc
var nameOfType = type.LibraryTypeName;
var guid = type.Guid;
var authorOfTheVersion = type.Author;
var status = type.State;
}
...
```

Modify the following program code to read Status Attribute on "User Folder" from Project Library:

```
...
ILibrary myProjectLibrary = Project.ProjectLibrary;
LibraryTypeUserFolder userFolder = myProjectLibrary.TypeFolder.Folders.Find("folder1");
var userTypeFolderConsistencyStatus = userFolder.Status;
if(userTypeFolderConsistencyStatus == ConsistencyStatus.DefaultVersionInConsistent)
{
//Do something.
}
...
```

Modify the following program code to read Status Attribute on "Individual Type" from Project Library:

```
...
ILibrary myProjectLibrary = Project.ProjectLibrary;
LibraryTypeUserFolder userFolder = myProjectLibrary.TypeFolder.Folders.Find("folder1");
ILibraryType blockType = userFolder.Types.Find("block1");
var blockTypeConsistencyStatus = blockType.Status;
if(blockTypeConsistencyStatus == ConsistencyStatus.DefaultVersionInConsistent)
{
//Do something.
}
...
```

Note

This attribute can also be used in similar way in Global Library.

Exception Handling

An ArgumentNullException exception will be thrown to you if any of the selected types or folders are null or not part of the source library.

```
var projectLibrary = Project.ProjectLibrary;
var type1 = projectLibrary.Types.FindType("Type1");
var status = typeFolder.Status; // throws ArgumentNullException
```

See also

Opening a project (Page 128)

5.6.32 Managing alarm text list in PLC and Mastercopy library

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to copy PLC text lists to a library as Master copy or delete PLC text lists from library Master copy folders. Both project and global libraries can be used.

You can also use the TIA Portal Openness to support the following functions :

- Accessing user text lists and systemtext lists
- Updating user text lists
- Deleting user text lists
- Setting text lists name and ListRange
- Accessing information about a textlist including ID, name, and type for user or system

Property

Reference of PlcAlarmTextlistGroup (Navigators)

Custom type	Navigator	Data type	Description
PlcUnitBase	PlcAlarmText-listGroup	PlcAlarmText-listGroup	It is the first level of text lists and creates the EOM classes of System and User text lists.
PlcSoftware			

Navigators of PlcAlarmTextlistGroup

Navigators	Datatype
PlcAlarmSystemTextlists	PlcAlarmSystemTextlistComposition
PlcAlarmUserTextlists	PlcAlarmUserTextlistComposition

PlcAlarmTextlist

The following properties are supported in PLC alarm text lists:

Property name	Data type	Description	Access
ID	Int32	Get id of the PLC alarm text lists	Read
ListRange	PlcAlarmTextlistLis-tRange	Get types of range for PLC alarm text lists. The PlcAlarmTextlistListRange can accept the following enum values: • Decimal: Decimal value of a range type • Binary: Binary value of a range type • Bit: Bit value of a range type • None: None value of a range type	Read
Name	String	Get name of the PLC alarm text lists	Read

Navigator of PlcAlarmTextlist

Navigator	Data type
Comment	MultilingualText

PlcAlarmUserTextlist

Property name	Data type	Description	Access
ListRange	PlcAlarmTextlistLis-tRange	Get the types of range for PLC alarm text lists	Read/Write
Name	String	Get the name of the PLC alarm text lists	Read/Write

Program code

```
TiaPortal tia = TiaPortal.GetProcesses().Select(x => x.Attach()).First();
Project project = tia.Projects.First();
Device device = project.Devices.First();
DeviceItem deviceItem = (DeviceItem)(from x in device.DeviceItems where
x.Name.Equals("PLC_1") select x).First();
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<SoftwareContainer>();
PlcSoftware plc = softwareContainer.Software as PlcSoftware;
var group = plc.PlcAlarmTextlistGroup;
var groupParent = group.Parent;
var systemTextlist = group.PlcAlarmSystemTextlists.First();
var listRangeType = systemTextlist.ListRange;
var systemTextlistParent = systemTextlist.Parent;
var userTextlist = group.PlcAlarmUserTextlists.First();
var userTextlistParent = userTextlist.Parent;
var listRange = systemTextlist.ListRange;
var comment = systemTextlist.Comment;
userTextlist.Name = "set text";
userTextlist.ListRange =
Siemens.Engineering.SW.Alarm.TextLists.PlcAlarmTextlistListRange.Bit;
userTextlist.Comment.Items.Find(project.LanguageSettings.ActiveLanguages.Find(new
CultureInfo("en-US", false))).Text = "EN comment";
userTextlist.Comment.Items.Find(project.LanguageSettings.ActiveLanguages.Find(new
CultureInfo("de-DE", false))).Text = "DE comment";
userTextlist.Delete();
MasterCopy masterCopy =
project.ProjectLibrary.MasterCopyFolder.MasterCopies.First<MasterCopy>(x => x.Name ==
"textlist_in_library");
PlcAlarmUserTextlists.CreateFrom(masterCopy);
masterCopy.Delete();
project.ProjectLibrary.MasterCopyFolder.MasterCopies.Create(userTextlist);
```

5.7 Functions for accessing devices, networks and connections

5.7.1 Open the "Devices & networks" editor

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

You can open the "Devices & networks" editor via the API interface by using one of two methods:

- `ShowHwEditor(View.Topology` or `View.Network` or `View.Device)`: Open the "Devices & networks" editor from the project.
- `ShowInEditor(View.Topology` or `View.Network` or `View.Device)` : Displays the specified device in the "Devices & networks" editor.

Use the `View` parameter to define the view that is displayed when you open the editor:

- `View.Topology`
- `View.Network`
- `View.Device`

Program code

Modify the following program code to open the "Devices & networks" editor:

```
// Open topology view from project
private static void OpenEditorDevicesAndNetworksFromProject(Project project)
{
    project.ShowHwEditor(Siemens.Engineering.HW.View.Topology);
}
```

Modify the following program code to open the "Devices & networks" editor for a device:

```
// Open topology view for given device
private static void OpenEditorDevicesAndNetworksFromDevice(Device device)
{
    device.ShowInEditor(Siemens.Engineering.HW.View.Topology);
}
```

See also

Importing configuration data (Page 1212)

5.7.2 Querying PLC and HMI targets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

You can determine whether a software base can be used as PLC target (PlcSoftware) or HMI target in the TIA Portal Openness API.

Program code: PLC target

Modify the following program code to determine if a device item can be used as PLC target:

```
// Returns PlcSoftware
private PlcSoftware GetPlcSoftware(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
        if (softwareContainer != null)
        {
            Software softwareBase = softwareContainer.Software;
            PlcSoftware plcSoftware = softwareBase as PlcSoftware;
            return plcSoftware;
        }
    }
    return null;
 }
```

Program code: HMI target

Modify the following program code to determine if a device item can be used as HMI target:

```
//Checks whether a device is of type hmitarget
private HmiTarget GetHmiTarget(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
        if (softwareContainer != null)
        {
            Software softwareBase = softwareContainer.Software;
            HmiTarget hmiTarget = softwareBase as HmiTarget;
            return hmiTarget;
        }
    }
    return null;
}
```

See also

Enumerating devices (Page 339)

5.7.3 Accessing attributes of an address object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- For writing access, the PLC is offline.

Application

You can use the TIA Portal Openness API interface to get or set attributes of the address object.

Further you can assign the current process image to an OB.

The following attributes can be accessed:

Attribute name	Data type	Writeable	Access	Description
`IsochronousMode`	Bool	r/w	Dynamic attribute	Activate/Deactivate isochronousMode
`ProcessImage`	Int32	r/w	Dynamic attribute	Set/Get process image partition number.
`InterruptObNumber`	Int64	r/w	Dynamic attribute	Set/Get interrupt organization block number. (classic controller only)
`StartAddress`	Int32	r/w	Modelled attribute	Set/Get new StartAddress value.

Restriction

- Attribute `StartAddress`
  - Setting `StartAddress` may implicit change the `StartAddress` of the opposite I0 Type at the name module. Changing of input address changes the output address.
  - Writing access is not supported for all devices.
  - Packed addresses are not supported in TIA Portal Openness
  - Changing an address via TIA Portal Openness will not rewire the assigned tags.
- Attribute `InterruptObNumber`
  - Only accessible in settings with S7-300 or S7-400 controllers. Writing access is supported for S7-400 controllers.

Program code: Get or set attributes of an address object

Modify the following program code to access isochronous mode of an address object:

```
Address address= ...;

  // read attribute
bool attributeValue = (bool)address.GetAttribute("IsochronousMode");

 // write attribute
address.SetAttribute("IsochronousMode", true);
```

Modify the following program code to access the `ProcessImage` attribute of an address object:

```
Address address= ...;

  // read attribute
int attributeValue = (int)address.GetAttribute("ProcessImage");

 // write attribute
address.SetAttribute("ProcessImage", 7);
```

Modify the following program code to access the `InterruptObNumber` attribute of an address object:

```
Address address= ...;

  // read attribute
long attributeValue = (long)address.GetAttribute("InterrruptObNumber");

 // write attribute
address.SetAttribute("InterrruptObNumber", 42L);

//default value = 40
```

Modify the following program code to access the `StartAddress` attribute of an address object:

```
Address address= ...;

  // read attribute
int attributeValue = (int)address.GetAttribute("StartAddress");

 // write attribute
address.StartAddress = IntValueStartAddress;
```

Program code: Assign the current process image to an OB

Modify the following program code to assign the current process image to an OB:

```
OB obX =…
Address address= ...;

// assign PIP 5 to obX

address.SetAttribute("ProcessImage", 5);

try
{
        address.AssignProcessImageToOrganizationBlock(obX);
}    catch(RecoverableException e) {
       Console.WriteLine(e.Message);
}

// remove this PIP-OB assignment

try
{
        address.AssignProcessImageToOrganizationBlock(null);
}    catch(RecoverableException e) {
       Console.WriteLine(e.Message);
}
```

5.7.4 Accessing the channels of a module

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Signal Modules like analog input modules usually have multiple channels within a single module. Usually, channels provide similar functionality multiple times, e.g. an analog input module with four channels can measure four voltage values at the same time.

To access all channels of a module, the Channels attribute of an device item is used.

Program code: Attributes of channels

Modify the following program code to access attributes of a channel:

```
DeviceItem aiModule = ...
ChannelComposition channels = aiModule.Channels;
foreach (Channel channel in channels)
{
    ... // Work with the channel
}
```

Program code: Identifying attributes

Modify the following program code to get the identifying attribute for each channel:

```
Channel channel = ...
int channelNumber = channel.Number;
ChannelType type = channel.Type;
ChannelIoType ioType = channel.IoType;
```

Program code: Accessing a single channel

Modify the following program code to use the identifying attributes to access a channel directly:

```
DeviceItem aiModule = ...
Channel channel = aiModule.Channels.Find(ChannelType.Analog, ChannelIoType.Input, 0);
... // Work with the channel
```

Channel types

Value	Description
ChannelType.None	The channel type invalid.
ChannelType.Analog	The channel type is analog.
ChannelType.Digital	The channel type is digital.
ChannelType.Technology	The channel type is technology.

Channel IO types

Value	Description
ChannelIOType.None	The channel IO type invalid.
ChannelIOType.Input	An input channel.
ChannelIOType.Output	An output channel.
ChannelIOType.Complex	Complex IO types, e.g. for technological channels.

5.8 Functions on networks

5.8.1 Creating a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Subnets can be created in two different ways:

- Create a subnet that is connected to an interface: The type of the interface, where the subnet is created, determines the type of the subnet
- Create a subnet not connected to an interface.

Program code: Create a subnet connected to a node

Modify the following program code to create a subnet:

```
Node node = ...;
Subnet subnet = node.CreateAndConnectToSubnet("NameOfSubnet");
```

The following type identifiers are used:

- System:Subnet.Ethernet
- System:Subnet.Profibus
- System:Subnet.Mpi
- System:Subnet.Asi

Program code: Create a subnet not connected to an interface

Modify the following program code to create a subnet:

```
Project project = ...;
SubnetComposition subnets = project.Subnets;

Subnet newSubnet = subnets.Create("System:Subnet.Ethernet", "NewSubnet");
```

The following type identifiers can be used:

- System:Subnet.Ethernet
- System:Subnet.Profibus
- System:Subnet.Mpi
- System:Subnet.Asi

5.8.2 Accessing subnets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

For several network related features, e.g. assigning interfaces to a subnet, you have to access subnets in the project. Typically, subnets are aggregated directly on project level.

Program code: Access all subnets of a project

Modify the following program code to access all subnets excluding internal subnets of a project:

```
Project project = ...
foreach (Subnet net in project.Subnets)
{
    ... // Work with the subnet
}
```

Program code: Access a specific subnet

Modify the following program code to access a specific subnet by it's name:

```
Project project = ...
Subnet net = project.Subnets.Find("PROFIBUS_1");
{
    ... // Work with the subnet
}
```

Attributes of a subnet

A subnet has the following attributes:

```
Subnet net = ...;
string name = net.Name;
NetType type = net.NetType;
```

Network types

Value	Description
NetType.Unknown	The type of the network is unknown.
NetType.Ethernet	The type of the network is Ethernet.
NetType.Profibus	The type of the network is Profibus.
NetType.Mpi	The type of the network is MPI.
NetType.ProfibusIntegrated	The type of the network is integrated Profibus.
NetType.Asi	The type of the network is ASi.
NetType.PcInternal	The type of the network is PC internal.
NetType.Ptp	The type of the network is PtP.
NetType.Link	The type of the network is Link.
NetType.Wan	The type of the network is Wide Area Network

5.8.3 Accessing internal subnets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

If a device item is able to compose a subnet, it provides the additional functionality "subnet owner". To access this additional functionality, a specific service of the device item must be used.

Program code: Get the subnet owner role

Modify the following program code to get the subnet owner role:

```
SubnetOwner subnetOwner =
((IEngineeringServiceProvider)deviceItem).GetService<SubnetOwner>();
if (subnetOwner != null)
{
    // work with the role
}
```

Program code: Attributes of a subnet owner

Modify the following program code to access the subnets of a subnet owner:

```
foreach(Subnet subnet in subnetOwner.Subnets)
{
    Subnet interalSubnet = subnet;
}
```

5.8.4 Get type identifier of subnets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The attribute `TypeIdentifier` is used to identify a subnet. The `TypeIdentifier` is a string consisting of several parts: `<TypeIdentifierType>:<SystemIdentifier>`

Possible values for `TypeIdentifierType` are:

- `System`

SystemIdentifier

Subnet type	SystemIdentifier
PROFIBUS	Subnet.Profibus
MPI	Subnet.Mpi
Industrial Ethernet	Subnet.Ethernet
ASI	Subnet.Asi
Ptp	Subnet.Ptp

Subnet type	SystemIdentifier
ProfibusIntegrated	Subnet.ProfibusIntegrated
PcInternal	Subnet.PcInternal
ProfidriveIntegrated	Subnet.ProfidriveIntegrated

Program code

Modify the following program code to get the type identifier for user manageable and separately creatable objects for GSD:

```
Subnet subnet = ...;
string typeIdentifier = subnet.TypeIdentifier;
```

5.8.5 Accessing attributes of a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

A subnet provides certain mandatory attributes that can be read and/or written. The attributes are only available, if they are available at the UI. Writing is generally only allowed if an attribute can also be changed by the user at the UI. This might vary depending on the type of the subnet. For example, the user can only set the DpCycleTime, if the IsochronousMode is true and the DpCycleMinTimeAutoCalculation is false

Attributes of subnets of type ASI

Attribute	Data type	Writable	Access	Description
Name	string	r/w	modeled	Name of the subnet.
NetType	NetType	r	modeled	Type of the subnet
SubnetId	string	r	dynamic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.

Attributes of subnets of type Ethernet

Attribute	Data type	Writable	Access	Description
Name	string	r/w	modeled	Name of the subnet.
NetType	NetType	r	modeled	Type of the subnet
SubnetId	string	r/w	dynamic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.
DefaultSubnet	bool	r/w	dynamic	true if the subnet is a default subnet. There is at most one default subnet in a project.

Attributes of subnets of type MPI

Attribute	Data type	Writable	Access	Description
Name	string	r/w	modeled	Name of the subnet.
NetType	NetType	r	modeled	Type of the subnet
SubnetId	string	r/w	dynamic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.
HighestAddress	int	r/w	dynamic	Highest MPI address at subnet.
TransmissionSpeed	BaudRate	r/w	dynamic	True if the subnet is a default subnet. There is at most one default subnet in a project.

Attributes of subnets of type PC internal

Attribute	Data type	Writable	Access	Description
Name	string	r	modeled	Name of the subnet.
NetType	NetType	r	modeled	Type of the subnet
SubnetId	string	r	dynamic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.

Attributes of subnets of type PROFIBUS

Attribute	Data type	Writable	Access	Description
Name	string	r/w	modeled	Name of the subnet.
NetType	NetType	r	modelled	Type of the subnet
SubnetId	string	r/w	dynamic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.
HighestAddress	int	r/w	dynamic	Highest PROFIBUS address at subnet.
TransmissionSpeed	BaudRate	r/w	dynamic	True if the subnet is a default subnet. There is at most one default subnet in a project.
BusProfile	BusProfile	r/w	dynamic	The PROFIBUS profile.
PbCableConfiguration	bool	r/w	dynamic	True to enable additional PROFIBUS network settings
PbRepeaterCount	int	r/w	dynamic	Number of repeaters for copper cable
PbCopperCableLength	double	r/w	dynamic	The length of the copper cable
PbOpticalComponentCount	int	r/w	dynamic	Number of OLMs and OBTs of fiber-optical cable.
PbOpticalCableLength	double	r/w	dynamic	The length of the fiber-optical cable for the PROFIBUS network in km.
PbOpticalRing	bool	r/w	dynamic	True if bus parameter are adapted for an optical ring
PbOlmP12	bool	r/w	dynamic	True if OLM/P12 is enabled for bus parameter calculation
PbOlmG12	bool	r/w	dynamic	True if OLM/G12 is enabled for bus parameter calculation
PbOlmG12Eec	bool	r/w	dynamic	True if OLM/G12-EEC is enabled for bus parameter calculation
PbOlmG121300	bool	r/w	dynamic	True if OLM/G12-1300 is enabled for bus parameter calculation
PbAdditionalNetworkDevices	bool	r/w	dynamic	True if additional bus devices that don't exist in project will be taken in to account when calculating bus times.
PbAdditionalDpMaster	int	r/w	dynamic	Number of unconfigured DP masters.
PbTotalDpMaster	int	r	dynamic	Number of total DP masters
PbAdditionalPassiveDevice	int	r/w	dynamic	Number of unconfigured DP slaves or passive devices.
PbTotalPassiveDevice	int	r	dynamic	Number of total DP slaves or passive devices.

Attribute	Data type	Writable	Access	Description
PbAdditionalActiveDevice	int	r/w	dynamic	Number of unconfigured active devices with FDL/FMS/S/ communication load.
PbTotalActiveDevice	int	r	dynamic	Number of total active devices with FDL/FMS/S/ communication load.
PbAdditionalCommunicationLoad	CommunicationLoad	r/w	dynamic	Rough quantification of the communication load
PbDirectDateExchange	bool	r/w	dynamic	Optimization for direct data exchange.
PbMinimizeTslotForSlaveFailure	bool	r/w	dynamic	Minimization for time allocation for slave failure.
PbOptimizeCableConfiguration	bool	r/w	dynamic	Optimiazation of the cable configuration.
PbCyclicDistribution	bool	r/w	dynamic	True if enable cyclic distribution of bus parameter.
PbTslotInit	int	r/w	dynamic	Default value of Tslot.
PbTslot	int	r	dynamic	Waiting to receive time (slot time)
PbMinTsdr	int	r/w	dynamic	Minimum protocol processing time
PbMaxTsdr	int	r/w	dynamic	Maximum protocol processing time
PbTid1	int	r	dynamic	Idle time 1
PbTid2	int	r	dynamic	Idle time 2
PbTrdy	int	r	dynamic	Ready time
PbTset	int	r/w	dynamic	Setup time
PbTqui	int	r/w	dynamic	Quiet time for modulator
PbTtr	int64	r/w	dynamic	The Ttr value in t_Bit
PbTtrTypical	int64	r	dynamic	Average response time on bus
PbWatchdog	int64	r/w	dynamic	Watchdog
PbGapFactor	int	r/w	dynamic	Gab update factor
PbRetryLimit	int	r/w	dynamic	Maximum number of retries
IsochronousMode	bool	r/w	dynamic	True if constant bus cycle time is enabled.
PbAdditionalPassivDeviceForIsochronousMode	int	r/w	dynamic	Number of additional OPs/PGs/TDs etc. that are not configured in this network view.

Attribute	Data type	Writable	Access	Description
PbTotalPassivDeviceForIsochronous-Mode	int	r	dynamic	Sum of configured and unconfigured devices, such as OPs/PGs/TDs etc.
DpCycleMinTimeAutoCalculation	bool	r/w	dynamic	True if automatic calculation and setting of shortest DP cycle time is enabled.
DpCycleTime	double	r/w	dynamic	The DP cycle time.
IsochronousTiToAutoCalculation	bool	r/w	dynamic	True if automatic calculation and setting of values of IsochronousTi and IsochronousTo.
IsochronousTi	double	r/w	dynamic	Time Ti (read in process values)
IsochronousTo	double	r/w	dynamic	Time To (output process values)

Attributes of subnets of type PROFIBUS Integrated

Attribute	Data type	Writable	Access	Description
Name	string	r/w	modeled	Name of the subnet.
NetType	NetType	r	modeled	Type of the subnet
SubnetId	string	r/w	dynamic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.
IsochronousMode	bool	r	dynamic	Enabled constant bus cycle time.
DpCycleMinTimeAutoCalculation	bool	r/w	dynamic	True if automatic calculation and setting of shortest DP cycle time is enabled.
DpCycleTime	double	r/w	dynamic	The DP cycle time.
IsochronousTiToAutoCalculation	bool	r/w	dynamic	True if automatic calculation and setting of values of IsochronousTi and IsochronousTo.
IsochronousTi	double	r/w	dynamic	Time Ti (read in process values)
IsochronousTo	double	r/w	dynamic	Time To (output process values)

Attributes of subnets of type PROFIdrive Integrated

Attribute	Data type	Writable	Access	Description
Name	string	r/w	dynamic	Name of the subnet
NetType	NetType	r	modelled	Type of the subnet
IsochronousMode	bool	r	dynamic	Enabled constant bus cycle time

Attribute	Data type	Writable	Access	Description
SourceCycleTime*	Int32	r/w	dynamic	The bus source cycle time
DpCycleTime	double	r/w	dynamic	The DP cycle time
IsochronousTiToAutoCalculation	bool	r/w	dynamic	True if automatic calculation and setting of values of Ti (read in process values) and To (output process values)
IsochronousTi	double	r/w	dynamic	Time Ti (read in process values)
IsochronousTo	double	r/w	dynamic	Time To (output process values)

The *SourceCycleTime supports the following Enums value:

Enum Name	Value
Manual	0
AutomaticMinimum	1
LocalSendClock	2
ProfinetSendClock	3

Program code

Modify the following program code to get or set the attributes of a subnet:

```
SubnetOwner subnetOwner =
((IEngineeringServiceProvider)deviceItem).GetService<SubnetOwner>();
Subnet subnet = subnetOwner.Subnets[0];
string nameValue = subnet.Name;
NetType nodeType = (NetType)subnet.NetType;
string subnetId = ((IEngineeringObject)subnet).GetAttribute("SubnetId");
subnet.Name = "NewName";
subnet.SetAttribute("Name", "NewName");
bool isDefaultSubnet = ((IEngineeringObject)subnet).GetAttribute("DefaultSubnet");
```

Baud rates

Value	Description
BaudRate.None	The baud rate is unknown.
BaudRate.Baud9600	9.6 kBaud
BaudRate.Baud19200	19.2 kBaud
BaudRate.Baud45450	45.45 kBaud
BaudRate.Baud93700	93.75 kBaud
BaudRate.Baud187500	187.5 kBaud
BaudRate.Baud500000	500 kBaud
BaudRate.Baud1500000	1.5 MBaud
BaudRate.Baud3000000	3 MBaud
BaudRate.Baud6000000	6 MBaud
BaudRate.Baud12000000	12 MBaud

Bus profiles

Value	Description
BusProfile.None	The bus profile is unknown.
BusProfile.DP	The type of the network is DP.
BusProfile.Standard	The type of the network is Standard.
BusProfile.Universal	The type of the network is Universal.
BusProfile.UserDefined	The type of the network is user defined.

Communication load

Value	Description
CommunicationLoad.None	No valid communication load.
CommunicationLoad.Low	Typically used for DP, no great data communication apart from DP.
CommunicationLoad.Medium	Typically used for mixed operations featuring DP and other communication services, such as for S7 communication, when DP has strict time requirements and for average acyclic volumes of communication.
CommunicationLoad.High	For mixed operations featuring DP and other communication services, such as for S7 communication, when DP has loose time requirements and for high acyclic volumes of communication.

5.8.6 Deleting a global subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to delete a a global subnet within a project.:

```
Project project = ...;
SubnetComposition subnets = project.Subnets;
subnets.First();
// delete subnet
Subnet subnetToDelete = ...;
subnetToDelete.Delete();
```

5.8.7 Enumerate all participants of a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Enumeration of all participants on a subnet.

Program code

Modify the following program code to enumerate dp master systems from subnet:

```
Subnet subnet = ...;
foreach (Node node in subnet.Nodes)
    {
        // work with the node
    }
```

5.8.8 Enumerate IO systems of a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Enumeration of IoSystem provides all io systems that are present on a subnet. The class IoSystem represents the master systems and the io systems.

Program code

Modify the following program code to enumerate dp master systems from subnet:

```
Subnet subnet = ...;
foreach (IoSystem ioSystem in subnet.IoSystems)
    {
        // work with the io system
    }
```

5.8.9 Accessing nodes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The role interface aggregates nodes to access attributes that are related to the address and subnet assignment of an interface.

The name of a node can be seen in the attributes of an interface in TIA Portal . The NodeId is a unique identifier for every node aggregated at an interface, its value can only be seen via TIA Portal Openness.

Program code

Modify the following program code to access all nodes of an interface:

```
NetworkInterface itf = ...
foreach (Node node in itf.Nodes)
{
    ... // Work with the node
}
```

Most interfaces provide only a single node, therefore, usually the first node is used:

```
NetworkInterface itf = ...
Node node = itf.Nodes.First();
{
    ... // Work with the node
}
```

Nodes provide their names and types and NodeIds as attributes:

```
Node node = ...
string name = node.Name;
NetType type = node.NodeType;
string id = node.NodeId;
```

5.8.10 Accessing attributes of a node

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

A device item provides certain mandatory attributes that can be read and/or written. The attributes are only available if they are available at the UI. Writing is generally only allowed if an attribute can also be changed by the user at the UI. This might vary depending on the type of the device item. For example, the user can only set the RouterAddress if the RouterUsed is true. If the user changes the SubnetMask at IO controller, Subentmask on all IO devices will be also changed to the same value.

Attributes of a node of type ASI

Attributes	Data type	Writable	Access	Description
Name	string	r	modeled	Name of the node.
NodeId	string	r	modeled	ID of the node.
NodeType	NetType	r	modeled	A node gets his type from the subnet.
Address	string	r/w	dynamic	Additional attribute for AS-i slaves.

Attributes of a node of type Ethernet

Attributes	Data type	Writable	Access	Description
Name	string	r	modeled	Name of the node.
NodeId	string	r	modeled	ID of the node.
NodeType	NetType	r/w sometimes r/o	modeled	A node gets his type from the subnet.
UseIsoProtocol	bool	r/w	dynamic	True, if ISO protocol should be used
MacAddress	string	r/w	dynamic	e.g. 01-80-C2-00-00-00
UseIpProtocol	bool	r/w	dynamic	This value can be read even if it noct visible at the corresponding TIA UI control.
IpProtocolSelection	enum	r/w	dynamic	-
Address	string	r/w	dynamic	only IPv4 and no IPv6 is supported
SubnetMask	string	r/w	dynamic	-
UseRouter	bool	r/w	dynamic	-
RouterAddress	string	r/w	dynamic	-
DhcpClientId	string	r/w	dynamic	-
PnDeviceNameSetDirectly	bool	r/w	dynamic	PROFINET device name is set directly at the device. Not available for every device.
PnDeviceNameAutoGeneration	bool	r/w	dynamic	PROFINET device name is created automatically.
PnDeviceName	string	r/w	dynamic	Unique device name in subnet.
PnDeviceNameConverted	string	r	dynamic	Device name converted for system internal using.

Attributs of a node of type MPI

Attribut	Data type	Writable	Access	Description
Name	string	r	modeled	Name of the node.
NodeId	string	r	modeled	ID of the node.

Attribut	Data type	Writable	Access	Description
NodeType	NetType	r	modeled	A node gets his type from the subnet.
Address	string	r/w	dynamic	-

Attributs of a node of type PC internal

Attribut	Data type	Writable	Access	Description
Name	string	r	modeled	Name of the node.
NodeId	string	r	modeled	ID of the node.
NodeType	NetType	r	modeled	A node gets his type from the subnet.

Attributs of a node of type PROFIBUS

Attribut	Data type	Writable	Access	Description
Name	string	r	modeled	Name of the node.
NodeId	string	r	modeled	ID of the node.
NodeType	NetType	r	modeled	A node gets his type from the subnet.
Address	string	r/w	dynamic	The network address of the node. The type of address depends on the node type (e.g. IP address for PROFINET nodes, PROFIBUS address for PROFIBUS nodes)

Attributs of a node of type PROFIBUS Integrated

Attribute	Data type	Writable	Access	Description
Name	string	r	modeled	Name of the node.
NodeId	string	r	modeled	ID of the node.
NodeType	NetType	r	modeled	A node gets his type from the subnet.
Address	string	r	dynamic	-

Attributs of a node of type PROFIdrive Integrated

Attribute	Data type	Writable	Access	Description
Address	string	r	dynam-ic	The network address of the node.
ConnectedSubnet	Subnet	r	model-led	The connected subnet.
Name	string	r	model-led	Name of the node.
NodeId	string	r	model-led	ID of the node.
NodeType	NetType	r	model-led	A node gets its type from the subnet.

Program code: Attributes of a node

Modify the following program code to get or set the attributes of a node:

```
Node node = ...;
string nameValue = node.Name;
NetType nodeType = (NetType)node.NodeType;
node.NodeType = NetType.Mpi;
```

Program code: Dynamic attributes

Modify the following program code to get or set dynamic node attributes:

```
Node node = ...;
var attributeNames = new[]
{
    "Address", "SubnetMask", "RouterAddress", "UseRouter", "DhcpClientId",
"IpProtocolSelection"
};
foreach (var attributeName in attributeNames)
{
    object attributeValue = ((IEngineeringObject)node).GetAttribute(attributeName);
}
```

Protocol selection

Value	Description
IpProtocolSelection.None	Error value
IpProtocolSelection.Project	IP suite configured within project.
IpProtocolSelection.Dhcp	IP suite managed via DHCP protocol. DHCP Client ID necessary.
IpProtocolSelection.UserProgram	IP suite set via FB (function block).

Value	Description
IpProtocolSelection.OtherPath	IP suite set via other methods, for example PST tool.
IpProtocolSelection.ViaIoController	IP suite set via IO Controller in runtime.

Net type

Value	Description
NetType.Asi	Net type is ASI.
NetType.Ethernet	Net type is Ethernet.
NetType.Link	Net type is Link.
NetType.Mpi	Net type is MPI.
NetType.PcInternal	Net type is PC internal.
NetType.Profibus	Net type is PROFIBUS.
NetType.ProfibusIntegrated	Net type is PROFIBUS Integrated.
NetType.ProfidriveIntegrated	Net type is PROFIdrive Integrated.
NetType.Ptp	Net type is PTP.
NetType.Wan	Net type is Wide Area Network (WAN).
NetType.Unknown	Net type is Unknow.

5.8.11 Connecting a node to a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to assign a node (device, interface) to a network:

```
Node node = ...;
Subnet subnet = ...;
node.ConnectToSubnet(subnet);
```

5.8.12 Disconnect a node from a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to disconnect a node (device, interface) from a network:

```
Node node = ...;
node.DisconnectFromSubnet();
```

5.8.13 Creating an IO system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

An io system is created by calling the action IoController.CreateIoSystem("name") on an object of the type IoController. In case name is null or String.Empty the default name will be used. The io controller is aquired by accessing the attribute IoControllers object on the NetworkInterface. The IoControllers navigator returns one IoController object.

Prerequisites for creating an io system:

- The interface of the io controller is connected to a subnet.
- The io controller has no io system.

Program code

Modify the following program code to create an io system:

```
using System.Linq;
//...
NetworkInterface interface12 = ...;
IoSystem ioSystem = ...;
// Interface is configured as io controller
if((interface12.InterfaceOperatingMode & InterfaceOperatingModes.IoController) != 0)
{
IoControllerComposition ioControllers = interface12.IoControllers;
IoController ioController = ioControllers.First();
if(ioController != null)
{
ioSystem = ioController.CreateIoSystem("io system");
}
}
```

5.8.14 Accessing the attributes of an IO system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The master system and the io system will both be represented by the class IoSystem.

Program code: Attributes of an io system

Modify the following program code to get the attributes of the IoSystem:

```
NetworkInterface itf = ...
foreach (IIoController ioController in itf.IoControllers)
{
    IoSystem ioSystem = ioController.IoSystem;
    int ioSystemNumber = ioSystem.Number;
    string ioSystemName = ioSystem.Name;
}
```

IntroductionProgram code: Subnet of an io system

Modify the following program code to navigate to the subnet the io system is assigned to:

```
Subnet subnet = ioSystem.Subnet;
```

5.8.15 Connecting an IO connector to an IO system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Use the action ConnectToIoSystem(IoSystem ioSystem) of IoConnector to connect a profinet or a DP IoConnector to an existing io system.

Use the action GetIoController to navigate to the remote IoController. For further information how to navigate to the local IoConnector and the io system see Get master system or IO system of an interface (Page 293).

Prerequisites:

- The IoConnector is not yet connected to an io system.
- The IoConnector interface is connected to the same subnet as the interface of the desired IoController.

Program code

Modify the following program code:

```
NetworkInterface itf = ...;
IoSystem ioSystem = null;
foreach (IoController ioController in itf.IoControllers)
{
ioSystem = ioController.IoSystem;
IoConnector ioConnector = itf.IoConnectors[0];
ioConnector.ConnectToIoSystem();
IoController ioController2 = ioConnector.GetIoController();
}
```

5.8.16 Get master system or IO system of an interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The service NetworkInterface provides the navigator IoControllers, each IoController in turn provides the navigator IoSystem. The class IoSystem represents the master systems and the io systems. The io device and the slave are both named io device.

- The IoControllers navigator returns IoController objects, if the network interface can have an io system. At the moment only one io controller will be returned.
- The IoConnectors navigator returns IoConnector objects, if the network interface can be connected to an io system as an io device. At the moment only one io connector will be returned.

Program code: Get the io system of the IoController

Modify the following program code to get the io system of the IoController:

```
NetworkInterface itf = ...
foreach (IoController ioController in itf.IoControllers)
{
    IoSystem ioSystem = ioController.IoSystem;
    // work with the io system
}
```

Program code: Get the io system of the IoConnector

Modify the following program code to get the io system of the IoConnector:

```
NetInterface itf = ...
foreach (IoConnector ioConnector in itf.IoConnectors)
{
    IoSystem ioSystem = ioConnector.ConnectedIoSystem;
    // work with the io system
}
```

5.8.17 Get an IO Controller

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Currently only configurations with one IoController are possible. An IoController does not provide any modelled attributes or actions.

Program code

Modify the following program code to get the io controller:

```
NetworkInterface itf = ...
foreach (IoController ioController in itf.IoControllers)
{
    // work with the io controller
}
```

5.8.18 Get an IO Connector

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

An IoConnector does provide the modelled attributes or actions.

The following attribute and actions are available at the IoConnector:

Actions

Action	Signature	Description
ConnectToIoSystem	void ConnectToIoSystem(Siemens.Engineering.HW.IoSystem ioSystem	To connect an IoConnector to an existing DP master system
DisconnectFromIoSystem	void DisconnectFromIoSystem()	To disconnect an IoConnector from an existing io system
GetIoController	Siemens.Engineering.HW.IoController GetIoController()	Returns the IO controller for this connector

Links

Link	Type	Access
ConnectedToIoSystem	Siemens.Engineering.HW.IoSystem	read-only

Program code

Modify the following program code to get the io connector:

```
NetworkInterface itf = ...
foreach (IoConnector ioConnector in itf.IoConnectors)
{
    // work with the IoConnector
}
```

5.8.19 Disconnecting an IO connector from an IO system or a DP mastersystem

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Use the action DisconnectFromIoSystem() of IoConnector to disconnect an IoConnector from an existing io system or an existing DP mastersystem.

For further information how to navigate to the local IoConnector and the io system see Get master system or IO system of an interface (Page 293).

Program code

Modify the following program code:

```
IoSystem ioSystem = ...;
IoConnector ioConnector = ...;

ioConnector.DisconnectFromIoSystem();
```

5.8.20 Accessing attributes of a DP mastersystem

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

A DP Mastersystem provides certain attributes that can be read and/or written. The attributes are only available if they are available at the TIA Portal. Writing is generally only allowed if an attribute can also be changed by the user at the TIA Portal. This might vary depending on the DP Master and the DP Slaves which are assigned to this DP Mastersystem.

Attributes of a dp mastersystem

Attribute	Data type	Writable	Access	Description
Name	string	r/w	property	-
Number	int	r/w	property	The property Number accepts values that cannot be set via UI. In this case the compile will fail.

Program code: Get attributes

Modify the following program code to get attributes:

```
IoSystem dpMastersystem = ...;
string name = dpMastersystem.Name;
int number = dpMastersystem.Number;
```

Program code: Set attributes

Modify the following program code to set attributes:

```
IoSystem dpMastersystem = ...;
dpMastersystem.Name ="myDpMastersystem"
dpMastersystem.Number=42;
```

5.8.21 Accessing attributes of PN Driver

Application

You can use the TIA Portal Openness to access the attributes in PROFINET Interface of PN Driver V2.2.

Access to attribute of PN Driver V2.2 submodule object

The following attributes can be accessed at the Interface options using TIA Portal Openness:

Openness Attribute	Type	ReadOnly
DeviceReplacementWithoutExhangeableMedium	Boolean	False
OverwriteDeviceNames	Boolean	False
IECV22LLDPMode	Boolean	False

The following attributes can be accessed at the Real time options using TIA Portal Openness:

Openness Attribute	Type	ReadOnly
CalculatedBandwidthForCyclicIoData	Int32	True
MaxBandwidthForCyclicIoData	Int32	True

The following attributes can be accessed at Isochronous mode options using TIA Portal Openness:

Openness Attribute	Type	ReadOnly
ApplicationCycle	Int64	False
AutomaticMinimum	Boolean	False
DelayTime	Double	False

Note

The value of DelayTime cannot be set, unless the value of the AutomaticMinimum is set to False beforehand.

5.8.22 Accessing attributes of a profinet io system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

An IO System provides certain attributes that can be read and/or written. The attributes are only available if they are available at the UI. Writing is generally only allowed if an attribute can also be changed by the user at the UI. This might vary depending on the IO Controller and the IO Devices which are assigned to this IO System.

Attributes of a PROFINET io System

Attribute	Data type	Writable	Access	Description
MultipleUseIoSystem	bool	r/w	dynamic	-
Name	string	r/w	property	-
Number	int	r/w	property	The attribute Number accepts values that cannot be set via UI. In this case the compile will fail.
UseIoSystemNameAsDeviceNameExtension	bool	r/w	dynamic	If MultipleUseIoSystem is set to TRUE, UseIoSystemNameAsDeviceNameExtension will be set to FALSE and write access is not possible.
MaxNumberIWlanLinksPerSegment	int	r/w	dynamic	-

Program code: Get attributes

Modify the following program code to get attributes

```
IoSystem ioSystem = ...;
string name = ioSystem.Name;
```

Program code: Set attributes

Modify the following program code to set attributes:

```
IoSystem ioSystem = ...;
ioSystem.Name = "IOSystem_1";
```

Program code: Get attributes with dynamic access

Modify the following program code to get the values of dynamic attributes:

```
IoSystem ioSystem = ...;
var attributeNames = new[]
{
    "MultipleUseIoSystem", "UseIoSystemNameAsDeviceNameExtension",
"MaxNumberIWlanLinksPerSegment"
};
foreach (var attributeName in attributeNames)
{
    object attributeValue = ((IEngineeringObject)ioSystem).GetAttribute(attributeName);
}
```

Program code: Set attributes with dynamic access

Modify the following program code to set the values of dynamic attributes:

```
IoSystem ioSystem = ...;
((IEngineeringObject)ioSystem).SetAttribute("MultipleUseIoSystem", true);
```

5.8.23 Deleting a DP master system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code: Deleting a PROFINET io system

Modify the following program code to delete a PROFINET io system:

```
IoController ioController = ...;
IoSystem ioSystem = ioController.IoSystem;

ioSystem.Delete();
```

5.8.24 Deleting a profinet io system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to delete a profinet io system:

```
IoController ioController = ...;
IoSystem ioSystem = ioController.IoSystem;
ioSystem.Delete();
```

5.8.25 Creating a DP master system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

A DP master system is created by calling the action CreateIoSystem(string nameOfIoSystem) on an object of the type IoController. The io controller is aquired by accessing the attribute IoControllers object on the NetworkInterface.

Prerequisites for creating a DP master system:

- The interface of the io controller is connected to a subnet.
- The io controller has no io system.

Program code

Modify the following program code to create a dp master system:

```
//...
NetworkInterface interface12 = ...;
IoSystem dpMasterSystem = ...;
// Interface is configured as master or as master and slave
if((interface12.InterfaceOperatingMode & InterfaceOperatingModes.IoController) != 0)
{
IoControllerComposition ioControllers = interface12.IoControllers;
IoController ioController = ioControllers.First();
if(ioController != null)
{
dpMasterSystem = ioController.CreateIoSystem("dp master system");
}
}
```

5.8.26 Accessing port interconnection information of port device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

NetworkPort provides the link ConnectedPorts which is an enumeration of ports to access all interconnected partner ports of a port.

It is only possible to interconnect ports which can also be interconnected in the TIA UI, e.g. it is not possible to interconnect two ports of the same Ethernet interface. Recoverable exception are thrown

- if there is already an interconnection to the same partner port
- when trying to interconnect two ports which cannot be interconnected
- when trying to create a second interconnection to a port which does not support alternative partners

Program code: Get the port interconnection

Modify the following program code to get the port interconnection information of a port device item:

```
NetworkPort port = ...;
foreach (NetworkPort partnerPort in port.ConnectedPorts)
{
    // work with the partner port
}
```

Program code: Create port interconnections

Modify the following program code:

```
NetworkPort port1 = ...;
NetworkPort port2 = ...;
port1.ConnectToPort(port2);

// port supports alternative partners
NetworkPort port1 = ...;
NetworkPort port2 = ...;
NetworkPort port3 = ...;
port1.ConnectToPort(port2);
port1.ConnectToPort(port3);
```

Program code: Delete port interconnection

Modify the following program code:

```
NetworkPort port1 = ...;
NetworkPort port2 = ...;
port1.DisconnectFromPort(port2);
```

5.8.27 Attributes of port inter-connection

Attributes for port inter-connections

TIA Portal Openness supports the following attributes for port inter-connections. If the attributes are available in UI, the corresponding attributes can also be accessed through TIA Portal Openness. If the user has access to modifiy the attributes in UI, then they can also be modified via TIA Portal Openness.

Attribute name	Data type	Writable	Access	Description
MediumAttachment-Type	MediumAttachment-Type	Read-only	Dynamic attribute	-
CableName	CabelName	Read-Write	Dynamic attribute	-
AlternativePartnerPorts	Boolean	Read-Write	Dynamic attribute	Only available if tool-changer functionality is supported.
SignalDelaySelection	SignalDelaySelection	Read-Write	Dynamic attribute	-
CableLength	CableLength	Read-Write	Dynamic attribute	-
SignalDelayTime	Double	Read-Write	Dynamic attribute	-

Enum values of port inter-connection attributes

The Enum MediumAttachmentType has following values.

Value	Description
MediumAttachmentType.None	Attachment type cannot be determined.
MediumAttachmentType.Copper	Attachment type is copper.
MediumAttachmentType.FiberOptic	Attachment type is fiber optic.

The Enum CableName has following value

Value	Description
CableName.None	No cable name is specified
CableName.FO_Standard_Cable_9	FO standard cable GP (9 µm)
CableName.Flexible_FO_Cable_9	Flexible FO cable (9 µm)
CableName.FO_Standard_Cable_GP_50	FO standard cable GP (50 µm)
CableName.FO_Trailing_Cable_GP	FO trailing cable / GP
CableName.FO_Ground_Cable	FO ground cable
CableName.FO_Standard_Cable_62_5	FO standard cable (62.5 µm)
CableName.Flexible_FO_Cable_62_5	Flexible FO cable (62.5 µm)
CableName.POF_Standard_Cable_GP	POF standard cable GP
CableName.POF_Trailing_Cable	POF trailing cable
CableName.PCF_Standard_Cable_GP	PCF trailing cable / GP
CableName.GI_POF_Standard_Cable	GI-POF standard cable
CableName.GI_POF_Trailing_Cable	GI-POF trailing cable
CableName.GI_PCF_Standard_Cable	GI-PCF standard cable
CableName.GI_PCF_Trailing_Cable	GI-PCF trailing cable
CableName.GI_POF_Standard_Cable	GI-POF standard cable
CableName.GI_POF_Trailing_Cable	GI-POF trailing cable

Value	Description
CableName.GI_PCF_Standard_Cable	GI-PCF standard cable
CableName.GI_PCF_Trailing_Cable	GI-PCF trailing cable

The Enum SignalDelaySelection has following values.

Value	Description
SignalDelaySelection.None	-
SignalDelaySelection.CableLength	CableLength is used to define the signal delay.
SignalDelaySelection.SignalDelayTime	SignalDelayTime is used to define the signal delay

The Enum CableLength has following values

Value	Description
CableLength.None	CableLength is not specified.
CableLength.Length20m	Cable length is 20m.
CableLength.Length50m	Cable length is 50m.
CableLength.Length100m	Cable length is 100m.
CableLength.Length1000m	Cable length is 1000m.
CableLength.Length3000m	Cable length is 3000m.

Attributes of port options

The attributes of port options are given below.

Attribute name	Data type	Writable	Access
PortActivation	bool	Read-Write	Dynamic attribute
TransmissionRateAnd-Duplex	TransmissionRateAnd-Duplex	Read-Write	Dynamic attribute
PortMonitoring	bool	Read-Write	Dynamic attribute
TransmissionRateAuto-Negotiation	bool	Read-Write	Dynamic attribute
EndOfDetectionOfAc-cessibleDevices	bool	Read-Write	Dynamic attribute
EndOfTopologyDiscov-ery	bool	Read-Write	Dynamic attribute
EndOfSyncDomain	bool	Read-Write	Dynamic attribute

The Enum TransmissionRateAndDuplex has following values.

Value	Description
TransmissionRateAndDuplex.None	-
TransmissionRateAndDuplex.Automatic	Automatic
TransmissionRateAndDuplex.AUI10Mbps	10 Mbps AUI
TransmissionRateAndDuplex.TP10MbpsHalfDuplex	TP 10 Mbps half duplex
TransmissionRateAndDuplex.TP10MbpsFullDuplex	TP 10 Mbps full duplex
TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex	async fiber 10Mbit/s half duplex mode
TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex	async fiber 10Mbit/s full duplex mode
TransmissionRateAndDuplex.TP100MbpsHalfDuplex	TP 100 Mbps half duplex
TransmissionRateAndDuplex.TP100MbpsFullDuplex	TP 100 Mbps full duplex
TransmissionRateAndDuplex.FO100MbpsFullDuplex	FO 100 Mbps full duplex
TransmissionRateAndDuplex.X1000MbpsFullDuplex	X1000 Mbps full Duplex
TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD	FO 1000 Mbps full duplex LD
TransmissionRateAndDuplex.FO1000MbpsFullDuplex	FO 1000 Mbps full Duplex
TransmissionRateAndDuplex.TP1000MbpsFullDuplex	TP 1000 Mbps full duplex
TransmissionRateAndDuplex.FO10000MbpsFullDuplex	FO 10000 Mbps full Duplex
TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	FO 100 Mbps full duplex LD
TransmissionRateAndDuplex.POFPCF100MbpsFullDuplexLD	POF/PCF 100 Mbps full duplex

5.8.28 Accessing the attributes of a port

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  SeeConnecting to the TIA Portal (Page 82)
- A profect is open.
  See Opening a project (Page 128)

Application

If a device item is a port, it provides additional functionality over a simple device item.

- It is possible to access the linked partner ports of the port
- It is possible to access the interface of the port

To access this additional functionality, the NetworkPort feature, a specific service of the device item, must be used.

Program code: Accessing a port

Modify the following program code to access attributes of a channel:

```
NetworkPort port = ((IEngineeringServiceProvider)deviceItem).GetService<NetworkPort>();
if (port != null)
{
    ... // Work with the port
}
```

Attributes of a port

A port has the following attributes:

```
NetworkPort port = ...;
var connectedPorts = port.ConnectedPorts;
var myInterface = port.Interface;
```

5.8.29 Enumerate DP master systems of a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Enumeration of IoSystem provides all DP mastersystems that are present on a subnet. The class IoSystem represents the master systems and the io systems.

Program code

Modify the following program code to enumerate DP master systems from subnet:

```
Subnet subnet = ...;
foreach (IoSystem ioSystem in subnet.IoSystems)
    {
        // work with the io system
    }
```

5.8.30 Enumerate assigned IO connectors

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The class IoSystem represents the master systems and the io systems.

It is used for:

- Enumeration of io connectors of a dp mastersystem
- Enumeration of io connectors of a profinet io system

Program code

Modify the following program code to enumerate assigned io connectors of the dp mastersystem:

```
IoSystem ioSystem = ...;
foreach (IoConnector ioConnector in ioSystem.ConnectedIoDevices)
    {
        // work with the io connector
    }
```

5.8.31 Connecting a DP IO connector to a DP master system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Use the action ConnectToIoSystem(IoSystem ioSystem) of IoConnector to connect an IoConnector to an existing DP mastersystem.

Use the action GetIoController to navigate to the remote IoController. For further information how to navigate to the local IoConnector and the io system see Get master system or IO system of an interface (Page 293).

Prerequisites:

- The IoConnector is not yet connected to an io system.
- The IoConnector interface is connected to the same subnet as the interface of the desired IoController.

Program code

Modify the following program code:

```
IoSystem ioSystem = ...;
IoConnector ioConnector = ...;
ioConnector.ConnectToIoSystem(ioSystem);
IoController ioController = ioConnector.GetIoController();
```

5.8.32 Accessing AS-i profile and parameter attributes for virtual slaves

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

The TIA Portal Openness supports the following additional AS-i profile parameters for the Virtual Slaves of the CTT5 AS-i Slave using the StructuredData names:

Name	Description
AsiProfileVirtualSlave1	This contains the AS-i profile parameters for the Virtual Slave 1
AsiProfileVirtualSlave2	This contains the AS-i profile parameters for the Virtual Slave2
AsiProfileVirtualSlave3	This contains the AS-i profile parameters for the Virtual Slave 3

The AS-i slave parameters for the Virtual slaves can be found below:

Name	Description
AsiParameterVirtualSlave1	This contains the AS-i parameter for the Virtual Slave 1
AsiParameterVirtualSlave2	This contains the AS-i parameter for the Virtual Slave 2
AsiParameterVirtualSlave3	This contains the AS-i parameter for the Virtual Slave 3

Program code

Modify the following program code to get and set the additional attributes of AS-i Slaves:

```
DeviceItem slaveModule = ...;
var structuredDataNamesProfile = new[]{"AsiProfileVirtualSlave1",
"AsiProfileVirtualSlave2",
"AsiProfileVirtualSlave3"};
var structuredDataNamesAsiParameter = new[]{"AsiParameterVirtualSlave1",
"AsiParameterVirtualSlave2",
"AsiParameterVirtualSlave3"};
var attributeNamesProfile = new[]{ "AsiProfileID", "AsiProfileIO", "AsiProfileID2",
"AsiProfileID1" };

string attributeNameAsiParameter = "AsiSlaveParameter";
foreach (var structuredDataName in structuredDataNamesProfile)
{
foreach (var attributeName in attributeNamesProfile)
{
StructuredData structuredData =
(StructuredData)slaveModule.GetAttribute(structuredDataName);

//get
UInt32 attributeValue = (UInt32)structuredData.GetAttribute(attributeName);
}
//set
slaveModule.SetAttribute(attributeName, (UInt32)5);
}
foreach (var structuredDataName in structuredDataNamesAsiParameter)
{
StructuredData structuredData =
(StructuredData)slaveModule.GetAttribute(structuredDataName);
//get
UInt32 attributeValue = (UInt32)structuredData.GetAttribute(attributeNameAsiParameter);
//set
slaveModule.SetAttribute(attributeName, (UInt32)5);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.8.33 Accessing CM DP as DP slave and transfer area

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to configure an ET200SP PLC as DP slave if a CM DP has been plugged. It is also possible in TIA Portal Openness to create, configure and delete transfer areas at the DP interface. The handling will be similar to the handling of transfer areas at PN interface described in Openness transfer areas for PnPn coupler (Page 317)

The following dynamic attributes are supported at the device item DP interface.

Attribute name	Data type	Access	Writeable
DpUseForTestCommissioningRouting	Boolean	only available if configured as DP slave	r/w
DpWatchdog	Boolean	only available if configured as DP slave and assigned to DP Master	r/w

Program code: Creating transfer areas

Modify the following program code to create a transfer area at the DP interface:

```
NetworkInterface cpuItf = CpuInterface.GetService<NetworkInterface>();
// Create TransferAreas
TransferAreaComposition transferAreas = cpuItf.TransferAreas;
// Simple TranferArea
TransferArea transferAreaExample = transferAreas.Create("Example TA MS",
TransferAreaType.MS);
```

Parameter of transfer areas

The following parameters are supported in transfer areas of DP slave configuration:

Parameter name	Data type	Access
Name	string	read/write
Direction	enum	read/write
Comment	string	read/write
LocalToPartnerLength	Int32	read/write
PartnerToLocalLength	Int32	read/write
LocalAddresses	AddressComposition object	read
PartnerAddresses	AddressComposition object	read
PositionNumber	Int32	read
Type	enum	read

Program code: Deleting transfer area

Modify the following program code to delete transfer area for DP slave configuration:

```
TransferArea transferAreaExample = transferAreas.Create("Example TA MS",
TransferAreaType.MS);
transferAreaExample.Delete();
```

See also

Opening a project (Page 128)

5.8.34 Coupling 1516 isochron central and decentral Profinet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to perform the following operations:

- Given a 1516 PLC with one isochron capable module and it is possible to set the PLC into isochronous mode
- Given the previous configuration where the module is assigned to OB61 and it is possible to set the module into isochronous mode
- Given the previous config with an additional ET200SP assigned to the PLC and a module configured in isochronous mode assigned to the same OB61 where the source of sendclock can be set (PLC and decentral are coupled, Source of sendclock is set to PN X1).

Attributes

Attribute name	Data type	Writeable	Available range/values	Prerequisites	Description	Access
IsochronousMode	bool	r/w	True/False	Isochronous capable device	get/set Isochronus mode	Dynamic attribute
SourceCycleTime	int	r/w	Device specific	Isochronous mode is enabled	get/set Source of sendclock	Dynamic attribute
SendClock	float	r/w	Device specific	Isochronous mode is enabled	get/set Send clock	Dynamic attribute
IsochronousTiToCalculationMode	enum	r/w	FromOB, AutomaticMinimum, Manual	Isochronous mode is enabled	get/set Ti/To values	Dynamic attribute
IsochronousTi	double	r/w	Device specific	Isochronous mode is enabled	get/set Time Ti	Dynamic attribute
IsochronousTo	double	r/w	Device specific	Isochronous mode is enabled	get/set Time To	Dynamic attribute

Program code: Examples with S71500/ET200MP station

Modify the following program code for PLC represents an S71500/ET200MP station:

```
//plc represents an S71500/ET200MP station
//Getting headmodule of the station
DeviceItem plcHeadModule = plc.DeviceItems[1];
//Getting and I/O module of the station
DeviceItem module = plc.DeviceItems[2];
```

Modify the following program code to set isochoronous mode properties on PLC headmodule:

```
//Turn on isochronous mode
plcHeadModule.SetAttribute("IsochronousMode", true);
//Set source of sendclock to "Use send clock of PROFINET interface [X1]"
plcHeadModule.SetAttribute("SourceCycleTime", 1);
//Set source of sendclock to "Local send clock"
plcHeadModule.SetAttribute("SourceCycleTime", 0);
//Set sendclock
plcHeadModule.SetAttribute("SendClock", (float)2.1);
//Set calculation mode to Automatic minimum
plcHeadModule.SetAttribute("IsochronousTiToCalculationMode",
IsochronousTiToCalculationMode.AutomaticMinimum);
//Set calculation mode to Manual
plcHeadModule.SetAttribute("IsochronousTiToCalculationMode",
IsochronousTiToCalculationMode.Manual);
//Set Time Ti
plcHeadModule.SetAttribute("IsochronousTi", (double)0.55);
//Set Time To
plcHeadModule.SetAttribute("IsochronousTo", (double)0.65);
```

Modify the following program code to set isochoronous mode properties on I/O modules:

```
//Isochronous relevant properties can be found on the Address of the submodule
DeviceItem subModule = module.DeviceItems[0];
Address address = subModule.Addresses[0];
//Turn on isochronous mode
address.SetAttribute("IsochronousMode", true);
//Set Process image to 3
address.SetAttribute("ProcessImage", 3);
```

Program code: SIMATIC Drive Controller

For Setting isochoronous mode properties on PLC headmodule, It is same as in case of S71500/ET200MP station.

Modify the following program code to set isochoronous mode properties of central module (X142):

Modify the following program code to set isochoronous mode properties of an integrated telegram:

5.8.35 Openness for CP 1604/CP 1616/CP 1626

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- To compile the project, all devices must be "offline".

Introduction

You can use the TIA Portal Openness application to configure transfer areas and transfer area mapping rules for the communication processors CP 1604/CP 1616 as of V2.8 (also as of V2.7 depending on the article number) and CP 1626 as of V1.1.

Configuration of transfer areas

Creating transfer areas

To create a "CD" type transfer area for a CP 1604 modify the following program code:

```
NetworkInterface cpItf = CP 1604Interface.GetService<NetworkInterface>();
//Create Transfer Areas
TransferAreaComposition transferAreas = cpItf.TransferAreas;
//Simple TranferArea of type Input
TransferArea transferAreaInput = transferAreas.Create("Input CD", TransferAreaType.CD);
```

The following attributes are used for creating transfer areas:

Attribute	Description	
name	Specifies the name of the transfer area to be created.	
type	Specifies the type of the transfer area to be created. The following types are possible:	
	TransferAreaType.CD	Data exchange controller device
	TransferAreaType.F_PS	Data exchange PROFIsafe
	TransferAreaType.TM	Transfer module mapping
	Note: It is not possible to change the type at a later date.	

Setting attributes of the transfer areas

To set attributes of a transfer area modify the following program code:

```
transferAreaTm.LocalToPartnerLength = 8;
transferAreaTm.Direction = TransferAreaDirection.LocalToPartner;
string name = transferAreaTm.Name
```

Some attributes must be set or queried, but all of them can be set or queried using the general calls "GetAttribute()" or "SetAttribute()" modify the following program code:

```
const string myIndividualComment = "MyIndividualComment";
transferAreaTm.SetAttribute("Comment", myIndividualComment);
Int32 updateTime = transferAreaTm.GetAttribute("TransferUpdateTime");
```

The following attributes can be set for transfer areas:

Attribute	Description	
Name (string)	Specifies the name of the transfer area.	
Direction	Specifies the direction in which the data of the transfer area is transferred. The following directions are possible:	
	TransferAreaDirection.LocalToPartner	Data of the transfer area is transferred from the IO device to the higher-level IO controller.
	TransferAreaDirection.partnerToLocal	Data of the transfer area is transferred from the higher-level IO controller to the IO device.
	TransferAreaDirection.bidirectional	Data of the transfer area can be transferred in both directions between the higher-level IO controller and the IO device. The "LocalToPartnerLength" attribute determines the length of the transferred data from IO device to the higher-level IO controller. The "PartnerToLocalLength" attribute determines the amount of data from the higher-level IO controller to the IO device
Comment (string)	Text box for a comment on the transfer area.	
LocalToPartnerLength	Specifies the data length of the transfer area that is transferred from the IO device to the higher-level IO controller.	
PartnerToLocalLength	Specifies the data length of the transfer area that is transferred from the higher-level IO controller to the IO device.	
LocalAdresses	Specifies the input and output addresses of the transfer area from the local device.	
PartnerAdresses	Specifies the input and output addresses of the transfer area in the higher-level IO controller.	
TransferUpdateTime(Int32)	Specifies the update time of the transfer area. Only set or queried for a transfer area of the type "TransferAreaType.TM".	
PositionNumber	Specifies the number of the virtual submodule of this transfer area.	
Type	Specifies the type of transfer area, read-only.	
TransferAreaMappingRules	Specifies the routing table of the routing area, read-only.	

Deleting transfer areas

To delete transfer areas modify the following program code:

```
transferAreaInput.Delete();
```

Iteration through transfer areas

To iterate transfer areas modify the following program code:

```
TransferAreaComposition transferAreas = cpItf.TransferAreas;
foreach (TransferArea transferArea in transferAreas)
{
transferArea.Delete();
}
```

The following attributes can be set for IO routing:

Attribute	Description
Offset	Bit-based offset within the routing area to which the data is to be assigned. The length of the offset is determined by the "Begin" and "End" attributes.
Target	Specifies the module or submodule of the IO device that contains the data to be assigned to the configuration of the IO device, a transfer area of the type "TM".
IoType	The “IoType” attribute can only be changed in a transfer area of the “Input” type. In addition, a mixed module must be configured as “Target” for this transfer area. Only then can you select whether the data of the inputs (IoType.Input) are to be read or whether the data of the outputs (IoType.Output) are to be read (back).
Begin	Specifies the beginning of the data to be read by the "Target" attribute.
End	Specifies the end of the data to be read by the "Target" attribute.

Configuration of IO routing

Creating IO routes

To create IO routes modify the following program code:

```
//Create TransferAreaMappingRule
TransferAreaMappingRuleComposition routingTable = transferArea.TransferAreaMappingRules;///
//Create a new IO route
TransferAreaMappingRule route1 = routingTable.Create();
```

Setting attributes of IO routes

The following attributes can be set for IO routing:

Attribute	Description
Offset	Bit-based offset within the routing area to which the data is to be assigned. The length of the offset is determined by the "Begin" and "End" attributes.
Target	Specifies the module or submodule of the IO device that contains the data to be assigned to the configuration of the IO device, a transfer area of the type "TM".
IoType	The “IoType” attribute can only be changed in a transfer area of the “Input” type. In addition, a mixed module must be configured as “Target” for this transfer area. Only then can you select whether the data of the inputs (IoType.Input) are to be read or whether the data of the outputs (IoType.Output) are to be read (back).
Begin	Specifies the beginning of the data to be read by the "Target" attribute.
End	Specifies the end of the data to be read by the "Target" attribute.

Deleting IO routes

To delete IO routes modify the following program code:

```
transferAreaMappingRule.Delete();
```

Iteration via IO routing

To iterate via IO routing modify the following program code:

```
TransferAreaMappingRuleComposition routingTable = transferArea.TransferAreaMappingRules;
foreach (TransferAreaMappingRule route in routingTable)
{
route.Delete();
}
```

5.8.36 Openness transfer areas for PnPn coupler

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- To compile the project, all device must be "offline"

Introduction

You can use the TIA Portal Openness to add module and submodule, delete and search transfer areas for PnPn coupler.

Configuration of transfer areas

Creating transfer areas

For example, to create module on the next free position, use the following program code:

```
NetworkInterface coupler_Itf = deviceItem.GetService<NetworkInterface>();
TransferAreaComposition transferAreas = coupler_Itf.TransferAreas;
// Create a transfer area of type input at next free positionnumber
TransferArea transferAreaInput = transferAreas.Create("Input", TransferAreaType.IN);
```

Note

It is not possible to create a submodule without Positionnumber.

For example, to create module on PositionNumber and submodule at the next free submodulepositionnumber at positionNumber, use the following program code:

```
// Create a transfer area of type input at positionnumber 5
TransferArea transferAreaInput = transferAreas.Create("Input", TransferAreaType.IN, 5);
```

The following attributes are used for creating transfer areas:

Attribute	Description	
Type	Specifies the type of the transfer area to be created. The following types are possible:	
	TransferAreaType.None	Default value
	TransferAreaType.IN	Transfer area Input
	TransferAreaType.OUT	Transfer area Output
	TransferAreaType.MSI	Transfer area Shared Input (MSI)
	TransferAreaType.MSO	Transfer area Shared Output (MSO)
	TransferAreaType.MSO_LOCAL	Transfer area Local Shared Output (MSO_LOCAL)
	TransferAreaType.RECORD_WRITE_STO	Transfer area for Record Data Write up to 8 data records will be buffered
	TransferAreaType.RECORD_WRITE_PUB	Transfer area for Record Data Write overwrites previous data record
	TransferAreaType.RECORD_READ_STO	Transfer area for Record Data Read reads the oldest data record in the buffer
	TransferAreaType.RECORD_READ_PUB	Transfer area for Record Data Read reads data record written last
	TransferAreaType.MSI_MSO	Transfer area Shared Input (MSI) / Transfer area Shared Output (MSO)
	TransferAreaType.IN_OUT	Transfer area Input / Transfer area Output
	TransferAreaType.LOCAL_RECORD_STO	Transfer area for local data record coupling, up to 8 data records are buffered
	TransferAreaType.LOCAL_RECORD_PUB	Transfer area for local data record coupling, overwrites previous data record
	TransferAreaType.SUB_MSI	Transfer area copy of Shared Input (MSI)
	TransferAreaType.SUB_MSO	Transfer area copy of Shared Output (MSO)
	TransferAreaType.SUB_LOCAL_RECORD_STO_READ	Transfer area for Record Data Read for reading the oldest data record in the buffer
	TransferAreaType.SUB_LOCAL_RECORD_PUB_READ	Transfer area for Record Data Read for reading the most recently written data record
	TransferAreaType.PROFISAFE_IN12_OUT6	Transfer area with 12 byte input and 6 byte output
	TransferAreaType.PROFISAFE_IN6_OUT12	Transfer area with 6 byte input and 12 byte output

Setting attributes of the transfer areas

To set attributes of a transfer area, use the following program code, for example:

```
// Change input length
CouplerTransf.LocalToPartnerLength = 5;
// Change output length
CouplerTransf.PartnerToLocalLength = 5;
// Change name
CouplerTransf.Name = "Testname";
```

Some attributes must be set or queried, but all of them can be set or queried using the general calls TransferArea.GetAttribute() or SetAttribute(). Use the following program code, for example

```
// Set Comment of Transferarea
string myIndividualComment = "MyIndividualComment";
CouplerTransf.SetAttribute("Comment", myIndividualComment);
// Get positionNumber of Transferarea
Int32 positionNumber = (Int32)CouplerTransf.GetAttribute("PositionNumber");
```

The following attributes can be set for transfer areas:

Attribute	Description	
Name (string)	Specifies the name of the transfer area	
Direction	Specifies the direction in which the data of the transfer area is transferred. The following directions are possible:	
	TransferAreaDirection.LocalTo- Partner	Data of the transfer area is transferred from the IO device to the higher-level IO controller.
	TransferAreaDirection.partnerTo- Local	Data of the transfer area is transferred from the higherlevel IO controller to the IO device.
	TransferAreaDirection.bidirectional	Data of the transfer area can be transferred in both directions between the higher-level IO controller and the IO device. The "LocalToPartnerLength" attribute determines the length of the transferred data from IO device to the higher-level IO controller. The "PartnerToLocal- Length" attribute determines the amount of data from the higher-level IO controller to the IO device
Comment (string)	Text box for a comment on the transfer area	
LocalToPartnerLength	Specifies the input data length of the transfer area	
PartnerToLocalLength	Specifies the output data length of the transfer area	
PositionNumber	Specifies the slot number of the transfer area It is not possible to create sub module without a position number	
ExtendedPositionNumber	Specifies the sub slot number of the transfer area The ExtendedPositionNumber is only required for TransferAreas with more than one submodule like MSI	
Type	Specifies the type of transfer area	
SharedDeviceAccessConfigured	Controls the access to PLC	
UpdateAlarm	Enable the update alarm of the transfer area	
RecordIndex	Set the data record number that you will be using when writing the data record	

Deleting transfer areas

To delete transfer areas, use the following program code:

```
TransferArea TransferAreaInput = transferAreas.Create("Input", TransferAreaType.IN);
TransferAreaInput.Delete();
```

Searching transfer areas

To get the TransferArea with posotionNumber and extendedPositionNumber, use the following program code:

```
// Find a transfer area with Positionnumber and ExtendedPositionNumber
TransferArea transferAreaFind = transferAreas.Find(4, 3);
```

Note

ExtendedPositionNumber is only needed for TransferAreas with more than one Submodule like MSI

To get the first TransferArea with this positionNumber, use the following program code:

```
// Find a transfer area with Positionnumber
TransferArea transferAreaFind = transferAreas.Find(5);
```

Note

If one of this attributes is not available, you will encounter an exception.

See also

Opening a project (Page 128)

5.8.37 Openness virtual modules/submodules for ET 200SP PN HF

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to add and delete virtual module/submodule for ET 200 SP PN HF.

The following property are supported in virtual modules:

Property name	Data type	EomAtomValue	EomAtom description
SharedDeviceAccessConfigured	Boolean		
Name	String		
PositionNumber	Int32		
VirtualType	UInt64	1	MsoLocal
AddSubModules	Int64		

The following property are supported in virtual submodules:

Property	Data type	EomAtom value	EomAtom Description
SharedDeviceAccessConfigured	Boolean		
Comment	String		
PositionNumber	Int32		
Name	String		
ActivateDataStatus	Boolean		
InputAddressLength	Int64		
OutputAddressLength	Int64		
VirtualSubType	UInt64	1	MsoLocal
		2	MsoSub

Program code: Create and delete virtual module

Modify the following program code to add a new module:

```
string Type = "OrderNumber:6ES7 155-6AU30-0CN0/V4.2";
Device ET200SP = newProject.Devices.CreateWithItem(Type, "ET200SP", "ET200SP");
DeviceItem Rack = ET200SP.DeviceItems.First();
string TypeIdentifier = "OrderNumber:ET200SP.Virtual.Module";
string Name = "VirtualIO_1";
int PositionNumber = 100;
DeviceItem VIM = Rack.PlugNew(TypeIdentifier, Name, PositionNumber);
```

Modify the following program delete a module:

```
DeviceItem Rack = ET200SP.DeviceItems.First();
string TypeIdentifier = "OrderNumber:ET200SP.Virtual.Module";
string Name = "VirtualIO_1";
int PositionNumber = 100;
DeviceItem VIM = Rack.PlugNew(TypeIdentifier, Name, PositionNumber);
VIM.Delete();
```

Program code: Create and delete virtual sub module

Modify the following program code to create virtual submodules in Openness:

```
VIM1.SetAttribute("AddSubModules", (Int64)1);
VIM2.SetAttribute("AddSubModules", (Int64)2);
```

Note

To add a submodule, you have to set the module property "AddSubModules" to count of added submodules. If you try to add more submodulesthan allowed, there is an exception.

Modify the following program code to delete a submodule:

```
SubModule3.Delete();
```

Note

You cannot delete the first two submodules.

5.8.38 Accessing domain settings

Requirements

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access the complete domain management. The following two new services are added to the Subnet class for accessing domain management via TIA Portal Openness:

- service MrpDomainOwner
- service SyncDomainOwner

Program code: Accessing MrpDomainOwner

The service MrpDomain provides the navigator MrpDomains which contains the MrpDomainComposition. The composition contains objects of type MrpDomain.

```
Subnet subnet = …;
MrpDomainOwner mrpDomainOwner = subnet.GetService<MrpDomainOwner>();
MrpDomainComposition mrpDomainComposition = mrpDomainOwner.MrpDomains;
```

Modify the following program code to create a new domain with a name newMrpDomain;

```
int countOfMrpDomains = mrpDomainComposition.Count;
MrpDomain someMrpDomain = mrpDomainComposition.ElementAt(3);
MrpDomain firstMrpDomain = mrpDomainComposition.First();
MrpDomain byName = mrpDomainComposition.Find("DomainName");
// create a new domain
MrpDomain newMrpDomain = mrpDomainOwner.MrpDomains.Create("newMrpDomain");
```

Modify the following program code to read/write attributes value of a MRP domain objects:

```
NetworkInterface toBeAdded = …;
newMrpDomain.SetAttribute("IsDefault", true);
newMrpDomain.SetAttribute("ManagerOutsideOfProjectActive", false);
var participants = firstMrpDomain.DomainParticipants;
int count = participants.Count;
participants.Add(toBeAdded);
foreach (NetworkInterface networkIf in participants)
{
// do something at the interface
}
newMrpDomain.Delete();
```

Program code: Accessing SyncDomainOwner

The service provides the navigator SyncDomains which contains the SyncDomainComposition. This composition contains objects of type SyncDomain.

```
Subnet subnet = …;
SyncDomainOwner syncDomainOwner = subnet.GetService<SyncDomainOwner>();
SyncDomainComposition syncDomainComposition = syncDomainOwner.SyncDomains;
```

Modify the following program code to create a new SyncDomain with specific name:

```
int countOfSyncDomains = syncDomainComposition.Count;
SyncDomain someSyncDomain = syncDomainComposition.ElementAt(3);
SyncDomain firstSyncDomain = syncDomainComposition.First();
SyncDomain byName = syncDomainComposition.Find("DomainName");
// create a new domain
SyncDomain newSyncDomain = syncDomainOwner.SyncDomains.Create("newSyncDomain");
```

Modify the following program code to read/write the attribute value of Sync Domain:

```
NetworkInterface toBeAdded = …;
string convertedName = newSyncDomain.ConvertedName;
newSyncDomain.SetAttribute("IsDefault", true);
newSyncDomain.SetAttribute("HighPerformanceActive", true);
newSyncDomain.SetAttribute("FastForwardingActive", true);
var participants = firstSyncDomain.DomainParticipants;
int count = participants.Count;
participants.Add(toBeAdded);
foreach (NetworkInterface networkIf in participants)
{
// do something at the interface
}
newSyncDomain.Delete();
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

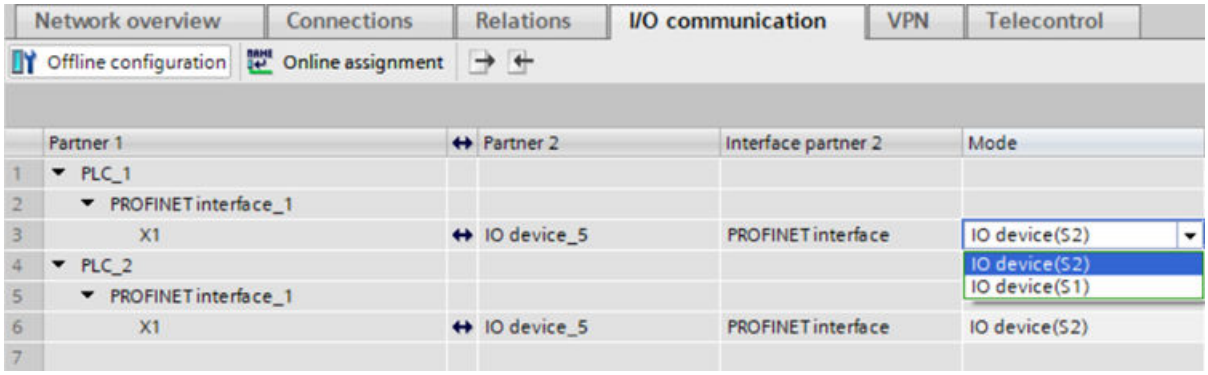
5.8.39 Accessing redundancy mode attribute

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal Openness to access an S2-capable device to be operated as a Switched S1 device. Using Openness, you should be able to change the Mode setting on the TIA Portal user interface for an IOD that is capable of system redundancy of type S2 to 'S1' as shown below:



The Mode property of TIA Portal is available in the TIA Portal Openness with name RedundancyMode. This attribute is available on S2 and R1 IO devices' Profinet interface.

The usage of this attribute is shown in the following table.

Attribute	Description
NetworkInterface.GetAttribute("RedundancyMode")	Reading the attribute via Openness
NetworkInterface.SetAttribute("RedundancyMode", "RedunancyMode.S2")	Setting the attribute via Openness
NetworkInterface.SetAttribute("RedundancyMode", "RedunancyMode.SwitchedS1")	

Program code

Modify the following program code to access redunancy mode attribute:

```
Device S2device = ...;
string s_S2DeviceInterface1Name = "";
NetworkInterface networkInterface1 = S2device.DeviceItems.First(f => f.Name ==
s_S2DeviceInterface1Name).GetService<NetworkInterface>();
string s_RedundancyMode = "...";
networkInterface1.SetAttribute(s_RedundancyMode, RedundancyMode.SwitchedS1);
//Assert
Assert.AreEqual(RedundancyMode.SwitchedS1,
networkInterface1.GetAttribute(s_RedundancyMode);
//Act
var systemBlockGroups = plc1?.GetService<ICompilable>();
CompilerResult compilerResult = systemBlockGroups?.Compile();
//Assert
Assert.AreEqual(0, compileResult?.ErrorCount);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.8.40 Creating CCDX transfer area

Requirement

- The application is connected to the TIA Portal via TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the Create() of the MulticastableTransferArea object in the TIA Portal Openness to create a CCDX transfer area. The Multicastable refers to a one to many relation that means a sender transfer area can cast information to multiple receiver partner transfer areas. Whereas in the opposite direction a receiver transfer area can communicate with only one sender partner.

Method

The overloading Create() accepts the following parameter name in Openness:

Parameter	Return type	Description
Create(NetworkInterface, TransferAreaType)	MulticastableTransferArea	Creates a Transfer Area and adds it to an existing CCDX connection with the specified type (e.g. DDX) on the specified network interface
Create(NetworkInterface, TransferAreaType, Name)	MulticastableTransferArea	Creates a Transfer Area by the sepcified name and adds it to an existing CCDX connection with the specified type (e.g. DDX) on the specified network interface
Create(NetworkInterface, TransferAreaType, Name, Length)	MulticastableTransferArea	Creates a Transfer Area by the sepcified name and length and adds it to an existing CCDX connection with the specified type (e.g. DDX) on the specified network interface.
Create(MulticastableTransfer-Area, TransferAreaType)	MulticastableTransferArea	Creates a new Transfer Area with the specified type and assigns it as receiver to an existing sender side transfer area.

Program code

```
// Given the NetworkInterfaces
private void CreatingTransferArea()
{
NetworkInterface senderNetworkInterface = ...;
NetworkInterface receiverNetworkInterface = ...;
MulticastableTransferArea transferArea1 =
senderNetworkInterface.MulticastableTransferAreas.Create(receiverNetworkInterface,
TransferAreaType.DDX);
string name = "Example_CCDX_TA";
MulticastableTransferArea transferArea2 =
senderNetworkInterface.MulticastableTransferAreas.Create(receiverNetworkInterface,
TransferAreaType.DDX, name);
int length = 32;
MulticastableTransferArea transferArea3 =
senderNetworkInterface.MulticastableTransferAreas.Create(receiverNetworkInterface,
TransferAreaType.DDX, name, length);
// Create receiver only
MulticastableTransferArea transferArea = ...;
MulticastableTransferArea transferArea4 =
senderNetworkInterface.MulticastableTransferAreas.Create(transferArea,
TransferAreaType.DDX);
}
```

See also

Opening a project (Page 128)

5.8.41 Deleting CCDX transfer area

Requirement

- The application is connected to the TIA Portal via TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the Delete() of the MulticastableTransferArea object in the TIA Portal Openness to delete the transfer area. The Multicastable refers to a one to many relation that means a sender transfer area can cast information to multiple receiver partner transfer areas. Whereas in the opposite direction a receiver transfer area can communicate with only one sender partner.

The Delete() behaves differently when deleting a receiver. There is always only 1 sender (One to Many).

Sender	Receiver	Behaviour
1	c = 1	Both the sender and the receiver will be deleted
1	c>1	Only the receiver will be deleted

The following behavior expected when deleting the sender:

Sender	Receiver	Behaviour
1	Any	Both the sender and all receivers will be deleted

Program code

```
private void DeleteTransferArea()
{
// Given the selected NetworkInterface
NetworkInterface networkInterface = ...;
// The name of the TransferArea
string transferAreaName = "Example_CCDX_TA";
networkInterface.MulticastableTransferAreas.FirstOrDefault(item => item.Name ==
transferAreaName).Delete();
}
```

See also

Opening a project (Page 128)

5.8.42 Modifying CCDX transfer areas configuration

Requirement

- The TIA Portal is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to modify the configuration of CCDX transfer areas. In Openness, you can access transfer area via NetworkInterface as TransferAreaMulticastableComposition objects which contain all the transfer areas that belong to the same interface of the same PLC .

The Multicastable refers to a one to many relation that means a sender transfer area can cast information to multiple receiver partner transfer areas. Whereas in the opposite direction a receiver transfer area can communicate with only one sender partner. (eg. CCDX transfer area).

Property

Property name	Data type	Description	Access
Address	AddressComposition	Gets the address information such as Start address, Length, IoType, etc. of a transfer area of type Siemens.Engineering.HW.Address	Read/Write
Parent	IEngineeringObject	Gets the EOM parent of a transfer area	Read-only
Comment	String	Gets the comment property of a transfer area	Read/Write
Direction	TransferAreaDirection	Gets the direction property of a transfer area as of type TransferAreaDirection. This indicates the direction of the communication depends on whether the transfer area is a sender or a receiver.	Read-only
DataLength	Int	Gets/sets the Data length (i.e. address length) of a transfer area	Read/Write
Name	String	Gets the name property of a transfer area	Read/Write
PartnerTransferAreas	MulticastableTransferAreaAssociation	Gets the partner transfer areas of a multicastable transfer area for eg. CCDX For example, sender transfer area contains its receiver transfer area partner(s) on the partner device (s). A sender transfer area can have multiple receiver partners. Both a sender or a receiver transfer area must have at least one partner, they cannot exist alone. Therefore, when deleting a sender transfer area, all it's receiver partners have to be deleted as well. But deleting a receiver transfer area doesn't necessarily come with the deletion of its sender partner if it has additional receiver partner.	Read-only
Type	TransferAreaType	Gets the Type property of a transfer area as of type TransferAreaType A multicastable transfer area can have below enum value: • TransferAreaType.DDX where DDX refers to transfer area type 'CCDX' for Direct Data Exchange communication.	Read-only
MulticastableTransferAreas	MulticastableTransferAreaComposition	Gets the MulticastableTransferAreas property of the networkinterface. It contains multicastable transfer areas (for eg. CCDX transfer areas) of a network interface. These can be both sender or receiver side transfer areas.	Read-only

Attribute	Data type	Description	Access
TransferUpdate-Time	Int32	Update time for the transmission of the provider (sender) item	Read-only

Program code

```
private void ModifyingCCDXTransferArea()
{
// Given the selected NetworkInterface
NetworkInterface networkInterface = ...;
// Navigate to the desired MulticastableTransferArea
string transferAreaName = "Example_CCDX_TA";
MulticastableTransferArea transferArea =
networkInterface.MulticastableTransferAreas.FirstOrDefault(item => item.Name ==
transferAreaName);
// Properties
Propertis properties = transferArea.Properties;
// Address
Address address = transferArea.Address;
// Parent
IEngineeringObject parent = transferArea.Parent;
// Comment
string comment = transferArea.Comment;
// Direction
TransferAreaDirection direction = transferArea.Direction;

// Name
string name = transferArea.Name;
// PartnerTransferAreas
MulticastableTransferAreaAssociation partnerTransferAreas =
transferArea.PartnerTransferAreas;
// TransferUpdateTime
int transferUpdateTime = (int)transferArea.GetAttribute("TransferUpdateTime");
// Type
TransferAreaType type = transferArea.Type;
// DataLength
transferArea.DataLength = 12;
}
```

5.9 Functions on devices

5.9.1 Mandatory attributes of devices

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Every device or device item provides certain mandatory attributes which can be read and/or written. These attributes are always the same as in the TIA Portal user interface.

The following attributes are supported in Openness:

Attribute name	Data type	Writeable	Access	Comment
Author	string	read/write	dynamic	
Comment	string	read/write	dynamic	sometimes only read access
CommentML	MultilingualTextItem	read/write	dynamic	sometimes only read access
IsGsd	bool	read		TRUE, if the device description is installed via GSD/GSDML
Name	string	read/write		sometimes only read access
TypeIdentifier	string	read		
TypeName	string	read	dynamic	

Program code: Mandatory attributes of a device

Modify the following program code to get the mandatory attributes of a device:

```
Device device = ...;
string nameValue = device.Name;
bool isGsdValue = device.IsGsd;
```

Program code: Mandatory attributes with dynamic access

Modify the following program code to get the attributes item with dynamic access:

```
Device device = ...;
var attributeNames = new[]
{
"TypeName", "Author", "Comment"};
foreach (var attributeName in attributeNames){object attributeValue =
((IEngineeringObject)device).GetAttribute(attributeName);
}
```

5.9.2 Get type identifier of devices and device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The attribute `TypeIdentifier` is used to identify a hardware object that is creatable via TIA Portal Openness API. The `TypeIdentifier` is a string consisting of several parts: `<TypeIdentifierType>:<Identifier>`

Possible values for `TypeIdentifierType` are:

- `OrderNumber`
- `GSD`
- `System`

OrderNumber

`OrderNumber` is the common `TypeIdentifier` for all modules present in the hardware catalog.

Format of type identifier	Example	Specifics
`<OrderNumber>`	OrderNumber:3RK1 200-0CE00-0AA2	-
`<OrderNumber>/<FirmwareVersion>`	OrderNumber:6ES7 510-1DJ01-0AB0/ V2.0	Firmware version is optional in case it does not exist or there is only one versionexisting in the system. Be care
`<OrderNumber>//` `<AdditionalTypeIdentifier>`	OrderNumber:6AV2 124-2DC01-0AX0// Landscape	The additional type identifier might be necessary in case that `OrderNumber` and `FirmwareVersion` do not lead to a unique match in the system.

Note

There are a few modules in the hardware catalog which use "wildcard" characters in the order number to represent a certain cluster of real hardware, e.g. the different lengths of S7-300 racks. In this case the specific `OrderNumber` and the "wildcard"-`OrderNumber` can both be used to create an instance of the hardware object. However you cannot generically use wildcards at any position.

GSD

This is the identifier used for modules that are added to the TIA Portal via GSD or GSDML.

Format of type identifier	Example	Specifics
`<GsdName>/<GsdType>`	GSD:SIEM8139.GSD/DAP	`GsdName` is the name of the GSD or GSDML in upper-case letters. `GsdType` is one of the following: • D: Device • R: Rack • DAP: HeadModule • M: Module • SM: Submodule GsdId is a identifier for the type.
`<GsdName>/<GsdType>/<GsdId>`	GSD:SIEM8139.GSD/M/4	

System

This is the identifier for objects, which cannot be determined using `OrderNumber` or `GSD`.

Format of type identifier	Example	Specifics
`<SystemTypeIdentifier>`	System:Device.S7300	`SystemTypeIdentifier` is the primary identifier of an object. `AdditionalTypeIdentifier` might be necessary in case the `SystemTypeIdentifier` is not unique. The prefix for certain object types are: • `Connection.` • `Subnet.` • `Device.` • `Rack.`
`<SystemTypeIdentifier>/` `<AdditionalTypeIdentifier>`	GSD:SIEM8139.GSD/M/4	

Program code

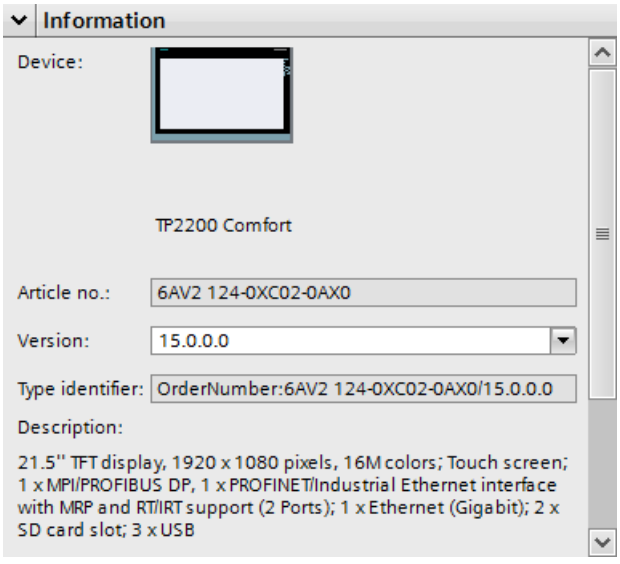
Modify the following program code to get the type identifier for user manageable and separately creatable objects for GSD:

```
HardwareObject hardwareObject = ...;
string typeIdentifier = hardwareObject.TypeIdentifier;
```

Displaying type identifiers in TIA Portal

If you need to know a type identifier you inquire it in TIA Portal as follows:

1. Enable the setting "Enable display of the type identifier for devices and modules" in "Options > Settings > Hardware configuration > Display of the type identifier".
2. Open the editor "Devices & networks".
3. Select a device in the Catalog.
   The type identifier is displayed in the viewlet "Information"



5.9.3 Set App ID on device and device Items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to set the Application ID on device and device items of TIA Portal project, so that these Application IDs are retained in the current session of TIA Portal for later use.

You can set the Application ID by providing "Application Key" and "Application Value" pair to the API.

The following constraints applicable for the Application ID of device and device item object:

- An object of type supporting this feature can have maximum of 64 Application IDs available for it
- The Application IDs Key or/and Value length can be maximum 128 Characters in length.
- The Application ID Key for a given object is unique
- Only one Application ID can be set using one "Application Key"
- When different Application Value is set with the same "Application Key", the last set Application Value is retained

Program code

Modify the following program code to set the Application ID for device and device item object:

```
Device device = ...;
// Ask for Service CustomIdentityProvider on the device/device item
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();
//Set the Application ID (Key-Value) pair
customIdentityProviderService.Set("Application_Key", "Application_Value");
```

5.9.4 Get App ID on device and device Items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Introduction

You can use the TIA Portal Openness to get the Application ID on device and device items of TIA Portal project, if the device or device items is already set with an Application ID.

Application IDs are stored as "Application Key" and "Application Value" pair as part of TIA Portal. You can use the TIA Portal Openness to get the Application ID value by providing "Application Key".

Program code

Modify the following program code to get the Application ID value for an object, which was already set using the Application Key:

```
Device device = ...;
// Ask for Service CustomIdentityProvider on the device/device item
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();
customIdentityProviderService.Set("Application_Key", "Application_Value");
//Get the Application ID (Value) for the given Application Key
var applicationValue = customIdentityProviderService.Get("Application_Key");
```

See also

Opening a project (Page 128)

5.9.5 Remove App ID on device and device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to remove Application ID (Custom Identity) for a device and device items of TIA Portal project, so that device and device item objects are updated with relevant Application ID.

A CustomIdentityNotFoundException will be thrown, if the Application ID which is passed as an argument does not exist.

Program code

Modify the following program code to remove the Application ID for device and device items:

```
Device device = ...;
// Ask for Service CustomIdentityProvider on the device/device item
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();
//Remove the CustomIdentity corresponding to Application ID
try
{
customIdentityProviderService.Remove("Application_Key");
}
catch(CustomIdentityNotFoundException ex)
{
// When CustomIdentity is not found for a the given key "Application_Key".
}
```

5.9.6 Creating a device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

A device can be created via two methods, within a project or a device group:

- Create a device via a device item type identifier like in TIA hardware catalog
  `Device CreateWithItem(DeviceItemTypeId, DeviceItemName, DeviceName)`
- Create only the device
  `Device Create(DeviceTypeId, DeviceName)`

Name	Type	Description
DeviceItemTypeId	string	Type identifier of the device item
DeviceTypeId	string	Type identifier of the device
DeviceItemName	string	Name of the created device item
DeviceName	string	Name of the created device

See: Get type identifier of devices and device items (Page 332)

Program code: Create device with type identifier

Modify the following code to create a device object via a type identifier:

```
DeviceComposition devices = ...;
Device device = devices.CreateWithItem("OrderNumber:6ES7 510-1DJ01-0AB0/V2.0", "PLC_1",
"NewDevice");
Device gsdDevice = devices.CreateWithItem("GSD:SIEM8139.GSD/M/4 ", "GSD Module",
"NewGsdDevice");
```

Program code: Create only the device

Modify the following code to create only the device object:

```
DeviceComposition devices = ...;
Device deviceOnly = devices.Create("System:Device.S7300", "S7300Device");
Device gsdDeviceOnly = devices.Create("GSD:SIEM8139.GSD/D", "GSD Device");
```

5.9.7 Enumerating devices

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

The TIA Portal Openness API positions devices similar to the project navigation in TIA Portal. PNV.

- Devices located as direct children of project are aggregated using the "Devices" composition of the project
- Devices located in device folders are aggregated using the "Devices" composition of the folder.

Note

Observe the Hierarchy of hardware objects of the object model (Page 65).

Use one of the following options to enumerate the devices of a project:

- Enumerate all devices at root level
- Enumerate all devices in groups or sub-groups

- Enumerate all devices of a project that contains no device groups
- Enumerate all devices of the ungrouped device system groups

Examples of devices that can be enumerated:

- Central station
- PB-Slave / PN-IO device
- HMI Device

Program code: Enumerating devices at root level

Modify the following program code to enumerate devices at root level:

```
private static void EnumerateDevicesInProject(Project project)
{
    DeviceComposition deviceComposition = project.Devices;
    foreach (Device device in deviceComposition)
    {
    // add code here
    }
}
```

Modify the following program code to access an individual device.

```
private static void AccessSingleDeviceByName(Project project)
{
    DeviceComposition deviceComposition = project.Devices;
    // The parameter specifies the name of the device
    Device device = deviceComposition.Find("MyDevice");
}
```

Program code: Enumerating devices in groups or sub-groups

To access devices in a group, you have to navigate to the group at first, after that to the device.

```
//Enamerate devices in groups or sub-groups
private static void EnumerateDevicesInGroups(Project project)
{
    foreach (DeviceUserGroup deviceUserGroup in project.DeviceGroups)
    {
        EnumerateDeviceUserGroup(deviceUserGroup);
    }
}
private static void EnumerateDeviceUserGroup(DeviceUserGroup deviceUserGroup)
{
    EnumerateDeviceObjects(deviceUserGroup.Devices);
    foreach (deviceUserGroup subDeviceUserGroup in deviceUserGroup.Groups)
    {
        // recursion
        EnumerateDeviceUserGroup(subDeviceUserGroup);
    }
}
private static void EnumerateDeviceObjects(DeviceComposition deviceComposition)
{
    foreach (Device device in deviceComposition)
    {
    // add code here
    }
}
```

Program code: Finding specific devices

Modify the following program code to find a specific device by name:

```
//Find a specific device by name
Project project = ...
Device plc1 = project.Devices.First(d => d.Name == "Mydevice");
... // Work with the device
```

Modify the following program code to find a specific device via "Find" method:

```
//Find a specific device via "Find" method
Project project = ...
Device plc1 = project.Devices.Find("MyDevice");
... // Work with the device
```

Program code: Enumerating devices of a project that contains no device groups

Modify the following program code:

```
//Enumerate all devices which are located directly under a project that contains no device
groups
Project project = ...
foreach (Device device in project.Devices)
{
    ... // Work with the devices
}
```

Program code: Enumerating all devices located in a folder

Modify the following program code:

```
//Enumerate all devices located in a folder
Project project = ...
DeviceUserGroup sortingGroup = project.DeviceGroups.Find ("Sorting");
Device plc1 = sortingGroup.Devices.First(d => d.Name == "MyStationName");
... // Work with the device
```

Program code: Enumerating devices of the ungrouped device system groups

To structure the projects, decentral devices have been put to the UngroupedDevices group. To access this group, navigate to the group at first, after that to the device.

Modify the following program code:

```
//Enumerate devices of the ungrouped device system group
Project project = ...
DeviceSystemGroup group = project.UngroupedDevicesGroup;
Device plc1 = group.Devices.First(d => d.Name == "MyStationName");
... // Work with the device
```

5.9.8 Accessing devices

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

Every GSD or GSDML based IO device has attributes. Some of them are used to identify the specific type of the device.

Name	Data type	Writeable	Access	Description
Author	string	read/write	dynamic	-
Comment	string	read/write	dynamic	-
GsdName	string	read	dynamic	Name of the GSD or GSDML file.
GsdType	string	read	dynamic	Type of the hardware object. For devices the value is always "D".
GsdId	string	read	dynamic	Specific identifier for the hardware object. For devices always empty.
IsGsd	bool	read	modeled	TRUE in case of a GSD device or a GSDML device
Name	string	read/write	modeled	-
TypeIdentifier	string	read	modeled	-

Program code: Get identification attributes

Modify the following program code to get the attributes:

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsdId"
    ;
foreach (var attributeName in attributeNames) {
    object attributeValue = device.GetAttribute(attributeName);
    }
```

Program code: Attributes

Modify the following program code to get the attributes:

```
Device device = ...;
string nameValue = device.Name;
bool isGsdValue = device.IsGsd;
```

Program code: Attributes with dynamic access

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsdId"
    ;
foreach (var attributeName in attributeNames) {
    object attributeValue = device.GetAttribute(attributeName);
    }
```

Program code: Specifics of GSD devices

If a device is a GSD device, it provides additional functionality. To get the GsdDevice feature, the GetService method is used.

```
GsdDevice gsdDevice = ((IEngineeringServiceProvider)deviceItem).GetService<GsdDevice>();
if (gsdDevice != null) {
    ... // work with the GSD device
    };
```

Program code: Attributes of a GSD device

Program code: Attributes of a GSD device

```
Device device = ...;
GsdDevice gsdDevice = ...;
string gsdId = gsdDevice.GsdId;
string gsdName = gsdDevice.GsdName;
string gsdType = gsdDevice.GsdType;
bool isProfibus = gsdDevice.IsProfibus;
bool isProfinet = gsdDevice.IsProfinet;
```

5.9.9 Accessing the TIA Portal hardware catalog

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can check manually required hardware in the catalog before running the Openness application. You can also use the TIA Portal Openns to respond correctly when a device is required to be added in the Openness application is not available in the hardware catalog for example - provide a message to the user that the GSD(ML) or HSP has to be installed first.

Attribute

A new type Siemens.Engineering.HW.HardwareCatalog.CatalogEntry is introduced which contains the following attribute members:

Attribute name	Data type	Description	Access
ArticleNumber	System.String	Specifies the article number of hardware object, for example. 6ES7 516-3AN00-0AB0	Read
CatalogPath	System.String	Specifies the catalog path of item for example: Root\PLC\SIMATIC S7-1500\CPU\CPU 1516-3 PN/DP\6ES7 516-3AN00-0AB0	Read
Description	System.String	Specifies the hardware catalog entry's description for example : • CPU with display; work memory 1 MB code and 5 MB data • 10 ns bit instruction time; • 4-stage protection concept, integrated technology functions: Motion Control, closed-loop control, counting and measuring • integrated tracing • 1st interface: PROFINET IO controller, supports RT/IRT, 2 ports, MRP, transport protocol TCP/IP, S7 communication, Web server, constant bus cycle time, routing • 2nd interface: PROFINET basic services, transport protocol TCP/IP, Web server, routing • 3rd interface: PROFIBUS DP master, constant bus cycle time, routing; firmware V1.0	Read

Attribute name	Data type	Description	Access
TypeIdentifier	System.String	Specifies the type identifier of item. e.g.: OrderNumber:6ES7 516-3AN00-0AB0/V1.0	Read
TypeIdentifierNormalized	System.String	Specifies the normalized typeIdentifier of item. for example: OrderNumber:6ES7516-3AN00-0AB0/V1.0	Read
TypeName	System.String	Specifies the type name of the device for example.: CPU 1516-3 PN/DP	Read
Version	System.String	Specifies the hardware catalog entry's version. for example: V1.0	Read

Program code

The function with one parameter filtered with an empty string returns the full, available hardware catalog:

```
TiaPortal portal = TiaPortal.GetProcesses().First().Attach();
var itemList = portal.HardwareCatalog.Find(string.Empty);
```

The function with one parameter finds hardware catalog entries and can be filtered by a filter string:

```
TiaPortal portal = TiaPortal.GetProcesses().First().Attach();
var itemList = portal.HardwareCatalog.Find("1516");
var first = itemList.First();
Console.WriteLine(first.TypeIdentifier);
//proj.Devices.Create("OrderNumber:6ES7 516-2GN00-0AB0/V2.9", "PLC_1");
Console.ReadKey();
```

The function with two parameters filter the search result for compatible/pluggable devices filtered for a given TypeIdentifier:

```
TiaPortal portal = TiaPortal.GetProcesses().First().Attach();
var itemList = portal.HardwareCatalog.Find("1BL00", "OrderNumber:6ES7 516-2GN00-0AB0/
V2.9");
//The itemList contains all the  devices which are compatible/pluggable with the given
TypeIdentifier
//and filtered by the filter string. In this case one of the itemList elements is 6ES7
521-1BL00-0AB0/V1.0.
Console.ReadKey();
```

5.9.10 Deleting a device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to delete a device:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
namespace DeletingDevice
{
internal class Program
{
private static void Main(string[] args)
{
Project project = ...;
// Needs to be: UngroupedDevicesGroup
Device deviceToDelete = project.UngroupedDevices.Devices.Find("......");
// delete device
deviceToDelete.Delete();
}
}
}
```

5.9.11 Bulk changing hardware parameters

Requirement

- The TIA Portal is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to configure multiple attributes of a hardware object using single SetAttributes call.

The API also removes the concern about the order and dependencies of the attributes and their corresponding values when function is called.

Signature

SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes, AttributeDelegate errorHandler);

Note

This is an overload of the SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes) API.
From TIA Portal V19 on bulk changing hardware parameters with ordering works the same way with all overloaded “SetAttributes” methods.

Program code

```
using Siemens.Engineering;
...
{
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var project = tiaPortal.Projects.Open(new FileInfo(project));
IList<KeyValuePair<string, object>> attributesToSet = new List<KeyValuePair<string,
object>>
{
new KeyValuePair<string, object>("IsochronousTi", (double)0.1), new KeyValuePair<string,
object>("IsochronousTo", (double)0.3),
new KeyValuePair<string, object>("IsochronousTiToCalculationMode",
Siemens.Engineering.HW.IsochronousTiToCalculationMode.Manual),
new KeyValuePair<string, object>("IsochronousMode", true),
};
project.DeviceItems[0].SetAttributes(attributesToSet, MyErrorHandler));
private void MyErrorHandler(AttributeConfiguration attributeConfiguration)
{
//this means, if one of the attribute set fails, the mechanism won't stop
//just skip the problematic part and the other attributes will be set.
attributeConfiguration.CurrentSelection = AttributeChoiceSelection.Ignore;
}
}
```

5.10 Functions on device items

5.10.1 Mandatory attributes of device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Every device or device item provides certain mandatory attributes which can be read and/or written. These attributes are always the same as in the TIA Portal user interface.

The following attributes are supported in TIA Portal Openness:

Attribute name	Data type	Writable	Access	Comment
Author	string	read/write	dynamic	-
Classification	DeviceItemClassification	read	modeled	The classification the device item belongs to, flags enum, see Device item classification description below.
Comment	string	read/write	dynamic	sometimes only read access
CommentML	MultilingualTextItem	read/write	dynamic	sometimes only read access
FirmwareVersion	string	read	dynamic	-
InterfaceOperatingMode	InterfaceOperatingModes	read/write	dynamic	For device items that provides the feature `NetworkInterface`
InterfaceType	NetType	readwrite	dynamic	For device items that provides the feature `NetworkInterface`
IsBuiltIn	bool	read	modeled	FALSE for objects creatable by the user
IsGsd	bool	read	modeled	TRUE, if the device description is installed via GSD/GSDML
IsPlugged	bool	read	modeled	TRUE for devices that are plugged
Label	string	read	dynamic	For device items that provides the feature `NetworkPort` or `NetworkInterface`. If the interface or port has no label `Label` will be `String.Empty`.
LocationIdentifier	string	read/write	dynamic	-
Name	string	read/write	modeled	sometimes only read access
OrderNumber	string	read	dynamic	-
PlantDesignation	string	read/write	dynamic	-
PositionNumber	int	read	modeled	-
TypeIdentifier	string	read	modeled	-
TypeName	string	read	dynamic	The language independent type name. Optional for device items that are not manageable by the user as auto-created items or fixed sub-modules).
InstallationDate	DateTime	read/write	dynamic	-
AdditionalInformation	string	read/write	dynamic	-

Device item classification

Value	Description
Siemens.Engineering.HW.DeviceItemClassifications.None	No classification.
Siemens.Engineering.HW.DeviceItemClassifications.CPU	The device item is a CPU
Siemens.Engineering.HW.DeviceItemClassifications.HM	The device item is a head module.

Program code: Mandatory attributes of a device item

Modify the following program code to get the mandatory attributes of a device item:

```
DeviceItem deviceItem = ...;
string nameValue = deviceItem.Name;
string typeIdentfierValue = deviceItem.TypeIdentifier;
int positionNumberValue = deviceItem.PositionNumber;
bool isBuiltInValue = deviceItem.IsBuiltIn;
bool isPluggedValue = deviceItem.IsPlugged;
```

Program code: Mandatory attributes with dynamic access

Modify the following program code to get the attributes item with dynamic access:

```
Device device = ...;
var attributeNames = new[] {
    "TypeName", "Author", "Comment", "OrderNumber", "FirmwareVersion", "PlantDesignation",
"LocationIdentifier"
    };
foreach (var attributeName in attributeNames) {
    object attributeValue = ((IEngineeringObject)deviceItem).GetAttribute(attributeName);
    }

DeviceItem deviceItem = ...;
((IEngineeringObject)deviceItem).SetAttribute("Comment", "This is a comment.");
```

5.10.2 Creating and plugging a device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The action `PlugNew(string typeIdentifier, string name, int positionNumber)` of `HardwareObject` is used to

- create a new device item and plug it into an existing hardware object
- create a new sub device item, e.g. a submodule, and plug it into a device item

If the action was succseful it returns the created device item object, otherwise a recoverable exception will be thrown.

By using the action `CanPlugNew(string typeIdentifier, string name, int positionNumber)` you can determine if creating and plugging is possible. If executing is not possible the action returns `false`.

If the method returns true, the action might still fail for the following unforeseen reasons.

- a position number is already taken by another device item
- the current device item cannot be plugged at the position although it is free
- the container does not provide the position number
- the name of the device item is already taken by an existing device item in the same container
- the device item cannot be plugged into the container
- the device is online

The following table shows the needed method parameters:

Name	Type	Description
`typeIdentifier`	string	type identifier of the created device item
`name`	string	name of created device item
`positionNumber`	int	position number of the created device item

Program code

Modify the following program code to plug a device item into an exisitng hardware object:

```
HardwareObject hwObject = ...;
string typeIdentifier = ...;
string name = ...;
int positionNumber = ...;
if(hwObject.CanPlugNew(typeIdentifier, name, positionNumber))
{
    DeviceItem newPluggedDeviceItem = hwObject.PlugNew(typeIdentifier, name,
positionNumber);
}
```

Accessing module information

The TIA Portal Openness user can access information about the pluggable modules using ModuleInformationProvider object. The user can access

- the container types in which a specified module has to be plugged (ex. Device and Rack) using FindContainerTypes method.
- the available versions for a certain partly specified module using FindModuleTypes method.

Program code: Accessing ModuleInformationProvider object

```
Project project = ...;
HardwareUtilityComposition extensions = project.HwUtilities;
var result = extensions.Find("ModuleInformationProvider") as ModuleInformationProvider;
```

Program code: Accessing container types using FindContinerTypes method

FindContinerTypes method returns the continer types of a given module. The module is speicified uisng typeIdentifier parameter. The resulting list contains the TypeIdentifiers of all container types of the requested type. Usually this includes a Device and a Rack and the containers are given in their order of the hierarchy in the project, beginning with the Device.

Name of the parameter	Type	Description
`typeIdentifier`	string	type identifier of a device item.

NOTICE
This method works only for the modules visible in the network view.
This method works only for modules, and not for sub-modules.

```
string typeIdentifier = ...;
string[] containerTypes = moduleInformationProvider.FindContainerTypes(typeIdentifier);
```

Program code: Accessing versions using FindModuleTypes method

FindModuleTypes method returns all possible versions of a hardware object using partial type identifier of the device item. This method returns a list of strings. Each string will be the complete TypeIdentifier of a possible match for the partial TypeIdentifier.

A valid partial type identifier must contain only complete parts, where each complete part is separated by "/" in the type identifier. Wildcards or incomplete parts are not supported. Also the following constraints with respect to the minimal number of specified parts must be observed:

- OrderNumber: at least one part. Ex. `OrderNumber:6ES7 317-2EK14-0AB0`
- GSD: at least two parts. Ex. `GSD:SI05816A.GSD/M`
- System: at lease one part. Ex. `System:Rack.ET200SP`

Name of the parameter	Type	Description
`partialTypeIdentifier`	string	partial type identifer of a device item

```
string partialTypeIdentifier = ...;
string[] moduleTypes = moduleInformationProvider.FindModuleTypes(partialTypeIdentifier);
```

Program code: Accessing plug locations using GetPlugLocations method

GetPlugLocations method returns the information about slots such as plug location, position number (designation of a slot), and available slots for the hardware object.

The class PlugLocation has the following peroperites.

Name of the property	Type	Description
`PositionNumber`	int	The position number of the free slot
`Label`	string	The label of the free slot

- In case no "label" exists for a certain position number, the string representation of the position number is used.
- `PlugLocation` objects are provided only for free slots.

```
HardwareObject hardwareObject = ...;
IList<PlugLocation> result = hardwareObject.GetPlugLocations();
foreach (PlugLocation item in result)
 {
    Console.WriteLine("{0} - {1}", item.PositionNumber, item.Label);
 }
```

5.10.3 Moving device items into another slot

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

The action PlugMove(DeviceItem deviceItem, int positionNumber) of HardwareObject is used to move an existing device item and plug it to an existing hardware object. The method PlugMove inserts the device items where the module was unable to plug in the UI. In these cases, there PlugMove action completes with complie errors.

The action CanPlugMove(DeviceItem deviceItem, int positionNumber) is used to determine possiblity of movement. If the movement is not possible, CanPlugMove returns false. If the method returns true, the action might still fail for the following unforeseen reasons.

- a position number is already taken by another device item
- the current device item cannot be plugged at the position although it is free
- the container does not provide the position number
- the name of the device item is already taken by an existing device item in the same container
- the device item cannot be plugged into the container
- the device item cannot be plugged by the user
- the device item cannot be removed by the user
- the device is online

Program code

Modify the following program code:

```
HardwareObject hwObject = ...;
DeviceItem deviceItemToMove = ...;
int positionNumber = ...;
if(hwObject.CanPlugMove(deviceItemToMove, positionNumber)
{
DeviceItem movedDeviceItem = hwObject.PlugMove(deviceItemToMove, positionNumber);
}
```

5.10.4 Copying a device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Use the action PlugCopy(DeviceItem deviceItem, int positionNumber) of HardwareObject to copy a device within a project and to plug it into an existing hardware. In rare cases the method PlugCopy might work where it is not possible to plug a module in the UI. In this case compile errors will occur after the copy. When PlugCopy was successful it returns the copy of the device item object, otherwise a recoverable exception is thrown.

Possible reasons for a failed action:

- a position number is already used by another device item
- the current device item cannot be plugged at the position although it is free
- the container does not provide the position number
- the name of the device item is already used by an existing device item in the same container
- the device item cannot be plugged into the container
- the device item cannot be plugged in the UI
- ...

Use the action CanPlugCopy(DeviceItem deviceItem, int positionNumber) is it possible to determine if copying should be possible. When it is not possible to execute the copy action CanPlugCopy returns false. However if the method returns true the action might still fail for unforeseen reasons.

Name of the parameter	Type	Description
deviceItem	DeviceItem	Device item to copy
positionNumber	int	position number for the copy of the device item

Program code

Modify the following program code:

```
HardwareObject hwObject = ...;
DeviceItem deviceItemToCopy = ...;
int positionNumber = ...;
if(hwObject.CanPlugCopy(deviceItemToCopy, positionNumber))
{
   DeviceItem copiedDeviceItem = hwObject.PlugCopy(deviceItemToCopy, positionNumber);
}
```

5.10.5 Setting user defined logo settings

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A Project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to set the user defined logo settings functionality in the CPU display of the S71500 PLC. The functionality is not supported by all of the S71500 PLCs.

The feature FrontPanel Display is available at the CPU display deviceitem to provide the following method :

Method name	Data type	Description
SetUserDefinedLogo()	FileInfo	Set the path for the image to be uploaded in the CPU display. The SetUserDefinedLogo() can only be used if the if the UserDefinedLogoActivated property is enabled. Otherwise, an EngineeringTargetInvocation exception will be thrown if the UserDefinedLogoActivated property is disabled.

The follwing properties can only be available in the Openness if the CPU contains a display and the properties are also available in the TIA Portal.

Property name	Data type	Description
UserDefinedLogoActivated	boolean	Displays the user defined logo. To upload the image in the CPU display, the UserDefinedLogoActivated property should be enabled. It is not possible to upload the image, if the UserDefinedLogoActivated property is disabled. Otherwise, you have to reset the other settings after disabling the UserDefinedLogoActivated property.
AdaptLogoActivated	boolean	Displays the adapt logo. The AdaptLogoActivated property is only available if the UserDefinedLogoActivated is enabled.

Program code

```
using Siemens.Engineering.HW.Features.FrontPanelDisplay;
private void UserDefinedLogoSettings()
{
FrontPanelDisplay frontPanelDisplay = plcDisplay.GetService<FrontPanelDisplay>();

//Enabling the user defined logo in the display
frontPanelDisplay.UserDefinedLogoActivated = true;

//Enabling the adapt logo in the display
frontPanelDisplay.AdaptLogoActivated = true;

//Uploading an image file in the display
frontPanelDisplay.SetUserDefinedLogo(new FileInfo(@"C:\Pictures\CompanyLogo.jpg"));
}
```

5.10.6 Deleting a device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to delete a device item:

```
Project project = ...;
var device = project.UngroupedDevicesGroup.Devices.Find("......");
var deviceItem = deviceItem.DeviceItems.First();
// delete device item
deviceItem.Delete();
```

5.10.7 Enumerate device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

To get to a device item use the HardwareObject. The items of a hardware objects are, what the user of the TIA Portal sees as being plugged into the hardware object:

- a rack which resides in a device
- a module which resides in a rack
- a sub module which resides in a module
- a sub module which resides in a sub module

Note

You can find more detailed information on this topic in the section Hierarchy of hardware objects of the object model (Page 65).

Program code: Enumerating device items of a device

Modify the following program code to enumerate device items of a hardware object:

```
private static void EnumerateDeviceItems(HardwareObject hardwareObject)
{
    foreach (DeviceItem deviceItem in hardwareObject.Items)
    {
        // add code here
    }
}
```

Program code: Enumerating with composition hierarchy

Modify the following program code if you want to enumerate the device items of a device by means of the composition hierarchy:

```
//Enumerates devices using an composition
private static void EnumerateDeviceItems(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        // add code here
    }
}
```

Program code: Enumerating devices items using an association

Modify the following program code to enumerate the device items using an association:

```
//Enumerates devices using an association
private static void EnumerateDeviceItemsWithAssociation(Device device)
{
    DeviceItemAssociation deviceItemAssociation = device.Items;
    foreach (DeviceItem deviceItem in deviceItemAssociation)
    {
        // add code here
    }
}
```

5.10.8 Accessing device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

To access objects of the type "DeviceItem" use the following attributes:

- Name (string): the name of the device item
- Container (HardwareObject): the container into which the device item is plugged

Name	Data type	Writeable	Access	Description
Author	string	read/write	dynamic	-
Comment	string	read/write	dynamic	-
FirmwareVersion	string	read	dynamic	Only for head modules
GsdName	string	read	dynamic	Name of the GSD file.
GsdType	string	read	dynamic	Type of the hardware object. For devices the value is always "D".
GsdId	string	read	dynamic	Specific identifier for the hardware object. For devices always empty.
IsBuiltIn	bool	read	-	-
IsGsd	bool	read	modeled	TRUE in case of a GSD device or a GSDML device
IsPlugged	bool	read	modeled	-
IsProfibus	bool	read	modeled	-
IsProfinet	bool	read	modeled	-
Name	string	read/write	modeled	-
OrderNumber	string	read	dynamic	Only for head modules
PositionNumber	bool	read	modeled	-
TypeIdentifier	string	read	modeled	-

Program code: Accessing a device item

Modify the following program code to access a device item:

```
public static DeviceItem AccessDeviceItemFromDevice(Device device)
{
    DeviceItem deviceItem = device.DeviceItems[0];
    return deviceItem;
}
```

Program code: Accessing a device item of a device item

Modify the following program code to access a device item of a device item:

```
public static DeviceItem AccessDeviceItemFromDeviceItem(DeviceItem deviceItem)
{
    DeviceItem subDeviceItem = deviceItem.DeviceItems[0];
    return subDeviceItem;
}
```

Program code: Navigating to the container of a device item

Modify the following program code to navigate back to the container of a device item via the "Container" attribute of DeviceItem:

```
DeviceItem deviceItem = ...;
HardwareObject container = deviceItem.Container;
```

Program code: Get identification attributes

Modify the following program code to get the attributes:

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsdId" };
foreach (var attributeName in attributeNames) {
    object attributeValue = ((IEngineeringObject)deviceItem).GetAttribute(attributeName);
    }
```

Program code: Get attributes

Modify the following program code to get the attributes:

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

string gsdName = gsdDeviceItem.GsdName;
string gsdType = gsdDeviceItem.GsdType;
string gsdId = gsdDeviceItem.GsdId;
bool isProfinet = gsdDeviceItem.IsProfinet;
bool isProfibus = gsdDeviceItem.IsProfibus;;
```

Program code: Get attributes with dynamic access

Modify the following program code to get the attributes:

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

var attributeNames = new[] {
    "TypeName", "Author", "Comment", ...
    ;
foreach (var attributeName in attributeNames) {
    object attributeValue =
((IEngineeringObject)gsdDeviceItem).GetAttribute(attributeName);
    }
```

Program code: Set attributes

Modify the following program code to set the attributes:

```
DeviceItem deviceItem = ...;
((IEngineeringObject)deviceItem).SetAttribute("Comment", "This is a
comment.");
```

Program code: Get prm data of a head module

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

int dsNumber = 0;       // For Profibus GSDs, dataset number zero must be used!
int byteOffset = 0;
int lengthInBytes = 5;

// read complete data set:
byte[] prmDataComplete = gsdDeviceItem.GetPrmData(dsNumber, byteOffset, lengthInBytes);

// read partial data set (only second byte):
byteOffset = 1;
lengthInBytes = 1;
byte[] prmDataPartial = gsdDeviceItem.GetPrmData(dsNumber, byteOffset, lengthInBytes);
```

Program code: Set prm data of a head module

Modify the following program code to get the prm data:

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

// The parameters byteOffset and the length of the byte array prmData define the range
within the
// dataset which is written to.
// For Profibus GSDs, dataset number zero must be used!

// Change the highlighted bytes 2-4 from 0x0 to 0x1
// to write only the first two bytes: byte[] prmData = {0x05, 0x21};

int dsNumber = 0;
int byteOffset = 0;
byte[] prmData = {0x05, 0x21, 0x01, 0x01, 0x01};

gsdDeviceItem.SetPrmData(dsNumber, byteOffset, prmData);
```

See also

Hierarchy of hardware objects of the object model (Page 65)

5.10.9 Accessing device item as interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

If a device item is an interface, it provides additional functionalities over a simple device item. Using this interface, the user can access the nodes and operation mode of the interface. Due to this functionality, the device item can used as IoDevice (Secondary) or IoController (Primary) by accessing the NetworkInterface feature (a specific Service of the device item).

The properities of the interface is accessed using enum InterfaceOperatingModes.

Value	Description
InterfaceOperatingModes.None	Default
InterfaceOperatingModes.IoDevice	Interface operation mode "IoDevice" (Slave).

Value	Description
InterfaceOperatingModes.IoController	Interface operation mode "IoController" (Master).
InterfaceOperatingModes.IoDevice or InterfaceOperatingModes.IoController	Interface operation made both of the above.

Program code: Accessing the network interface feature

Modify the following program code to get the network interface feature

```
Device deviceItem = ...;
NetworkInterface itf =
((IEngineeringServiceProvider)deviceItem).GetService<NetworkInterface>();
if (itf != null)
{
//Working with the interface
}
//Accessing nodes and operating mode
NodeComposition nodes = itf.Nodes;
InterfaceOperatingModes mode = itf.InterfaceOperatingMode;
//Accessing the type of interface
NetType itfType = itf.InterfaceType;
//Modififying the operating mode and interface type
itf.InterfaceOperatingMode = InterfaceOperatingModes.IoDevice;
itf.InterfaceType = NetType.Profibus;
//Accessing the ports linked to an interface.
NetworkPortAssociation nodes2 = itf.Ports;
```

5.10.10 Accessing change device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the ChangeType() API to change the type of device or device item. There are some limitations to the feature in the TIA Portal Openness as compared to the TIA Portal user interface:

- No list of possible change partners provided.
- No feedback message available to inform about the type changes to the changed device or device item.

The modules supported in the TIA Portal user interface can also be used in the TIA Portal Openness. The following functionalities are not supported in the feature of the TIA Portal Openness:

- GSDML exchange
- Module description update

The ChangeType() API fails to change the type of the device or device item for the following reasons:

- The given TypeIdentifier is unknown.
- The object to be changed is read-only.
- The type change is not supported, such as when switching to a completely different module type or family.

Program code

```
using Siemens.Engineering.HW.DeviceItem;
private void ChangeTypeDeviceItem()
{
DeviceItem rack = ...;
DeviceItem deviceItem = rack.PlugNew("OrderNumber:6ES7 516-3AN01-0AB0/V2.8", "PLC_1", 1);
try
{
deviceItem.ChangeType("OrderNumber:6ES7 516-3AN01-0AB0/V2.9");
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine("ChangeType failed: {0}", e.Message);
}
}
```

5.10.11 Accessing attributes of an I/O device interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- For writing access, the PLC is offline.

Application

You can use the TIA Portal Openness API interface to get or set attributes for IRT ans isochronous mode on the I/O device interface.

Access to the interface of an I/O controller

The following attributes can be accessed to the interface of an I/O controller. The controller has to be the sync master:

Attribute name	Data type	Writeable	Access	Description
PnSendClock	Int64	r/w	Dynamic attribute	Send clock in nanoseconds

Access to the interface of an I/O system

The following attributes can be accessed to the interface of an I/O system. The Ti/To values can be used by all modules and sub modules which belong to the I/O system.

Attribute name	Data type	Writeable	Access
IsochronousTiToAutoCalculation	BOOL	r/w	Dynamic attribute
IsochronousTi	DOUBLE	r/w	Dynamic attribute
IsochronousTo	DOUBLE	r/w	Dynamic attribute

Access to the interface of an I/O device

The following attributes can be accessed to the interface of an I/O device. The Ti/To values can be used by all modules and submodules which belong to the I/O sytem.

Attribute name	Data type	Writeable	Access
IsochronousMode	BOOL	r/w	Dynamic attribute
IsochronousTiToCalculationMode	IsochronousTiToCalculationMode	r/w	Dynamic attribute
IsochronousTi	DOUBLE	r/w	Dynamic attribute
IsochronousTo	DOUBLE	r/w	Dynamic attribute

The following ENUM values are provided for the attribute `IsochronousTiToCalculationMode`:

Value	Description
IsochronousTiToCalculationMode.None	-
IsochronousTiToCalculationMode.FromOB	Ti/To values of the OB (configured at the IoSystem) are used.
IsochronousTiToCalculationMode.FromSubnet	This value is not used by PROFINET interfaces.
IsochronousTiToCalculationMode.AutomaticMinimum	Ti/To values are calculated automatically for the IO Device.
IsochronousTiToCalculationMode.Manual	The user can enter Ti/To values for this IO Device manually.

Program code: Get or set attributes of an I/O device interface

Modify the following program code to access the send clock value:

```
DeviceItem pnInterface = ...;
// read attribute
long attributeValue = (long)pnInterface.GetAttribute("PnSendClock");
// write attribute
long sendClock = 2000000;
pnInterface.SetAttribute("PnSendClock", sendClock);
```

Modify the following program code to access the Ti/To values of OB:

```
IoSystem ioSystem = ...;
bool titoAutoCalculation = (bool)ioSystem.GetAttribute("IsochronousTiToAutoCalculation");
ioSystem.SetAttribute("IsochronousTiToAutoCalculation", true);
```

Modify the following program code to access the isochronous setting of an I/O device interface:

```
DeviceItem pnInterface = ...;
bool isochronousMode = (bool)pnInterface.GetAttribute("IsochronousMode");
pnInterface.SetAttribute("IsochronousMode", true);
```

5.10.12 Getting subnet of device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the NetworkInterface service to access the navigator Nodes. Each node in turn provides access to the subnet link by calling the ConnectedSubnet property.

Program code

To get to the subnet of a device or device item modify the following program code:

```
using Siemens.Engineering.HW.Node;
private void GetSubnetDeviceItem()
{
DeviceItem itfDeviceItem = ...;
NetworkInterface itf = itfDeviceItem.GetService<NetworkInterface>();
foreach (Node node in itf.Nodes)
{
Subnet subnet = node.ConnectedSubnet;
// work with the subnet
}
}
```

Note

In the above code the subnet is null if the node was not connected.

5.10.13 Accessing attributes of IoController

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- For writing access, the PLC is offline.

Application

You can use the TIA Portal Openness API interface to get or set attributes for IoController. The following attributes are only available at PROFINET IoController (located below a Profinet interface). If the user can modify an attribute in UI, then the user can set the corresponding attribute through TIA Portal Openness.

Attribute name	Data type	Type	Access	Description
SyncRole	SyncRole	Read-write	Dynamic attribute	-
PnDeviceNumber	int	Read-only	Dynamic attribute	In TIA Portal UI, this property is located at the ethernet node (PROFINET section)

The Synchronication role property is available in PROFINET interface of the TIA Portal UI. The Enum SyncRole has the following values.

Enum value	Numerical value
SyncRole.NotSynchronized	0
SyncRole.SyncMaster	1
SyncRole.SyncSlave	2
SyncRole.RedundantSyncMaster	4

Program code: Setting attributes of IoController

```
IoController ioController= ...;
SyncRole syncRole = (SyncRole)((IEngineeringObject)ioController).GetAttribute("SyncRole");
((IEngineeringObject)ioController).SetAttribute("SyncRole", SyncRole.SyncMaster);
```

5.10.14 Accessing attributes of IoConnector

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- For writing access, the PLC is offline.

Application

You can use the TIA Portal Openness API interface to get or set attributes for IoConnector. The following attributes are only available at PROFINET IoController (located below a Profinet interface). If the user can modify an attribute in UI, then the user can set the corresponding attribute through TIA Portal Oopenness.

There are four types of attributes such as update time attributes, watchdog time attributues, synchronization attributes and , device number attributes.

Update time attributes

The update time attributes are given below.

Attribute name	Data type	Type	Access	Description
PnUpdateTimeAutoCalculation	Boolean	Read-write	Dynamic attribute	If this attribute is true, then the update time is calculated automatically.
PnUpdateTime	Int64	Read-write	Dynamic attribute	Update time is measured in nano seconds.
PnUpdateTimeAdaption	Boolean	Read-write	Dynamic attribute	-

Watchdog time attributes

The watchdog time attributes are given below.

Attribute name	Data type	Type	Access	Description
PnWatchdogFactor	Int32	Read-write	Dynamic attribute	-
PnWatchdogTime	Int64	Read-only	Dynamic attribute	Watchdog time is measured in nano seconds.

Synchronization attributes

The synchronization attributes are given below.

Attribute name	Data type	Type	Access	Description
RtClass	RtClass	Read-write	Dynamic attribute	-
SyncRole	SyncRole	Read-only	Dynamic attribute	-

The Enum RtClass has following values.

Enum value	Numerical value
RtClass.None	0
RtClass.RT	1
RtClass.IRT	2

The Enum SyncRole has following values.

Enum value	Numerical value
SyncRole.NotSynchronized	0
SyncRole.SyncMaster	1
SyncRole.SyncSlave	2
SyncRole.RedundantSyncMaster	4

Device number attributes

The device number attributes are given below.

Attribute name	Data type	Type	Access	Description
PnDeviceNumber	int	Read-Write	Dynamic attribute	Indicates the device number.

Program code: Getting and setting attributes of IoConnector

```
IoConnector connector = ...

var attributeNames = new[] {
"PnUpdateTimeAutoCalculation", "PnUpdateTime", "PnUpdateTimeAdaption", "PnWatchdogFactor",
"PnWatchdogTime", "RtClass", "SyncRole"
};

foreach (var attributeName in attributeNames)
 {
      object attributeValue = ((IEngineeringObject)connector ).GetAttribute(attributeName);
 }

connector.SetAttribute("PnUpdateTimeAutoCalculation", true);
```

See also

Opening a project (Page 128)

5.10.15 Accessing address controller

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

If a device item is an address controller, it provides additional functionality. To access the registered addresses of the address controller the role AddressController is used.

Program code: Get the address controller

Modify the following program code to get the address controller role:

```
AddressController addressController =
((IEngineeringServiceProvider)deviceItem).GetService<AddressController>();
    if (addressController != null)
        {
            ... // work with the address controller
        }
```

Attributes of an address controller

The attributes of an address controller are:

- RegisteredAddresses

Modify the following program code to get the attributes of an address controller:

```
AddressController addressController = ...;
foreach (Address registeredAddress in addressController.RegisteredAddresses)
        {
            ...
        }
```

5.10.16 Accessing addresses

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Address objects are aquired via the composition link `Addresses` of a device item. The attribute `Addresses` returns a collection of type `AddressComposition` which can be enumerated.

Program code: Get an address of a device item

To get the address of a device item modify the following program code:

```
AddressComposition addresses = deviceItem.Addresses;
    foreach(Address address in addresses)
        {
            // work with the address
        }
```

Program code: Get an address of an io controller

To get the address of an io controller modify the following program code:

```
AddressComposition addresses = ioController.Addresses;
    foreach(Address address in addresses)
        {
            // work with the address
        }
```

Attributes

An address supports the following attributes:

Attribute name	Data type	Writeable	Access	Comment
AddressControllers	AddressControllerAssociation	read	-	-
Context	enum: AddressContext	read	dynamic	only for diagnosis addresses an for special device items
IoType	enum: AddressIoType	read	-	-
StartAdress	Int32	read/write	modeled	-
Length	Int32	read	-	-

Value	Description
AddressIoType.Diagnosis	The type of the address io is Diagnosis.
AddressIoType.Input	The type of the address io is Input.
AddressIoType.Output	The type of the address io is Output.
AddressIoType.Substitute	The type of the address io is Substitute.
AddressIoType.None	The type of the address io is mot specified.

Value	Description
AddressContext.None	The address context is not applicable.
AddressContext.Device	A device address context.
AddressContext.Head	A head address context.

Program code: Read attributes

Modify the following program code to get the attributes:

```
AddressControllerAssociation addressControllers = address.AddressControllers;
Int32 startAddress = address.StartAddress;
AddressIoType addressType = address.IoType;
Int32 adressLength = address.Length;
```

Program code: Write attributes

Modify the following program code to write the attributes:

```
Address addressControllers = ...;
address.StartAddress = intValueStartAddress;
```

Program code: Attributes with dynamic access

Modify the following program code to get the attributes:

```
Address address= ...;
object attributeValue = ((IEngineeringObject)address).GetAttribute("Context");
```

5.10.17 Accessing hardware identifiers

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

Hardware identifier objects are acquired from the following objects:

- Device
- DeviceItem
- IoSystem

The hardware identifier is represented by the class `HwIdentifier` and is accessed via the attribute `HwIdentifiers`.

Program code: Get the hardware identifier

To make HwIdentifier available modify the following program code:

```
var hwObject = ...;
foreach(HwIdentifier hardwareIdentifier in hwObject.HwIdentifiers)
{
// Work with the HwIdentifer
}
```

Attributes of a hardware identifier

```
HwIdentifierControllerAssociation controllers = hwIdentifier.HwIdentifierControllers;
Int64 Identifier = hwIdentifier.Identifier;
```

5.10.18 Accessing hardware identifier controller

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

If a device item is an hardware identifier controller, it is possible to access the registered hardware identifiers. To access these HwIdentifierController, a specific service of the device item, is used.

Program code: Get the hardware identifier controller

To get the HwIdentifierController modify the following program code:

```
HwIdentifierController hwIdentifierController =
((IEngineeringServiceProvider)deviceItem).GetService<HwIdentifierController>();
    if (hwIdentifierController != null)
        {
            ... // work with the hardware identifier controller
        }
```

Program code: Attributes of a hardware identifier controller

The attributes of an address controller are:

- `RegisteredHwIdentifiers`: The hardware identifier controllers where the hardware identifier is registered.

Modify the following program code to get the attributes of an address controller:

```
HwIdentifierController hwIdentifierController = ...;
HwIdentifierAssociation controllers = hwIdentifierController.RegisteredHwIdentifiers;
```

5.10.19 Accessing channels of device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

A channel is represented by the Channel class. Channels are aquired from a device item via the attribute Channels of the DeviceItem class. The attribute Channels returns an implementation of ChannelComposition which can be enumerated. If the device items has no channels, the attribute Channels returns an empty collection.

Mandatory attributes

A channel supports the following mandatory attributes:

Attribute name	Data type	Writeable	Access	Comment
IoType	ChannelIoType	read	modelled	-
Type	ChannelType	read	modelled	-
Number	Int32	read	modelled	-
ChannelAddress	Int32	read	dynamic	Address of the channel in bits
ChannelWidth	UInt32	read	dynamic	Width of the channel in bits

Program code: Get channels of device item

Modify the following program code to get the channels of a device item:

```
ChannelComposition channels = deviceItem.Channels
foreach(Channel channel in channels)
    {
        // work with the channel
    }
```

Program code: Mandatory attributes of a channel

Modify the following program code to get the channels of a device item:

```
Channel channel = ...;
int channelNumber = channel.Number;
ChannelType type = channel.Type;
ChannelIoType ioType = channel.IoType;
```

Program code: Get values of attributes with dynamic access

Modify the following program code to get the values of dynamic attributes:

```
Channel channel = ...;
Int32 channelAddress = (Int32)((IEngineeringObject)channel).GetAttribute("ChannelAddress");
UInt32 channelWidth = (UInt32)((IEngineeringObject)channel).GetAttribute("ChannelWidth");
```

Program code: Set value of a dynamic attribute

Modify the following program code to set the value of a writeable dynamic attribute:

```
Channel channel = ...;
((IEngineeringObject)channel).SetAttribute("AnAttribute", 1234);
```

5.10.20 Accessing normalized type identifiers

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the GetTypeIdentifierNormalized() in the TIA Portal Openness to get the normalized format of TypeIdentifier of a module. The module is specified using a TypeIdentifier.

The GetTypeIdentifierNormalized() supports the following property:

Property name	Data type	Description
typeIdentifier	String	Specifies the type identifier of a device item

Program code

```
using Siemens.Engineering.HW.Utilities;
private void ProvideTypeIdentifier(ModuleInformationProvider moduleInformationProvider)
{
string typeIdentifier = ...;
string normalizedTypeIdentifier =
moduleInformationProvider.GetTypeIdentifierNormalized(typeIdentifier);
}
```

5.10.21 Creating and exporting psc file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- Opening a Project
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness API to create .psc file and download a device into it. In order to create a.psc file and download a device configuration into it, you can use Export method from a service CardReaderPscProvider. The service exists in namespace Siemens.Engineering.HW.Utilities.

Program code

Modify the following program code to create and export a .psc file.

```
// preconditions
Device ipcDevice = myProject.Devices.Find("IPC427D");
FileInfo exportFile = new FileInfo(@"C:\Users\Ertan\Documents\Automation\PC system
configuration74.psc");
// get the card reader provider from hardware utilities
HardwareUtilityComposition utilities = myProject.HwUtilities;
HardwareUtility utility = utilities.Find("CardReaderPscProvider");
CardReaderPscProvider crp = (CardReaderPscProvider)utility;
// do the export
crp.Export(ipcDevice, exportFile);
```

Creating and openning a .psc file with separate commands are not supported. In case you give already existing .psc file as a parameter, the file will not be overwritten and exception will be thrown. If it is not an existing file, then this .psc file will be created and finally the download will be made.

You have to make sure that the project can be compiled without problems. Otherwise, exception will be thrown.

Note

Because of safety reasons, Export operation is not supported for f-activated devices (an exception will be thrown during export) in **versions V18 and below.**

5.10.22 Connection handling for extension racks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to fetch, add, and remove extension rack connections so that you can make use of TIA Portal Openness to implement extension rack connection support during CAx export/import.

Program code

```
ImConnection imConnection = portDeviceItem.GetService<ImConnection>();
imConnection.Connect(partnerport);
imConnection.Disconnect();
imConnection.GetPartnerPort();
var imConnectionOwner = imConnection.OwnedBy;
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11 Functions for accessing the data of a PLC device

5.11.1 Functions for downloading data to PLC device

5.11.1.1 Accessing system diagnostics properties for Plus PLCs

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to configure system diagnostics properties for Plus PLC 1500.

The following system diagnostic properties for PLC1500 are supported in TIA Portal Openness:

Property name	Data type	Description
UseFixedSystemDiagnosticsA-larmIds	boolean	This attribute controls specifies whether system diagnostics alarms should have always the same alarm IDs (which corresponds with their ALID), or if it doesn't matter which IDs those alarms will get when creating them. This attribute is available for every PLCs of type S7-1500, no matter which FW version. The default value for new TIA projects is TRUE, but FALSE for upgraded projects. When the value is TRUE, the client alarm IDs of the system diagnostics shall have a fixed definition in a reserved area of 512 in maximum. Switching from FALSE to TRUE in a TIA project which is already downloaded to PLC may cause that: • the PLC program must be recompiled • the online PLC has to be stopped to load the changed hardware configuration of the PLC • HMIs without automatic update enabled must be reloaded In the R/H system the setting of the attributes is synchronized between the two PLCs
SuppressDeactivatingSystem-DiagnosticsAlarms	boolean	This attribute controls specifies whether system diagnostics alarms about the device deactivation should be suppressed, or not. This attribute is available for every PLCs of type S7-1500 with FW version 3.0 or greater, so it is not visible/available for smaller FW versions. The default value is FALSE. When the value is TRUE, the device deactivation alarms are suppressed. • Diagnostic buffer entries are not suppressed at all, therefore you can check the state of all the devices using DOE. • In the R/H system the setting of the attributes is synchronized between the two PLCs

Program code

```
Siemens.Engineering.HW.DeviceItem plcItem = …;
// get value of "UseFixedSystemDiagnosticsAlarmIds" via openness interface
bool isUseFixedSystemDiagnosticsAlarmIds =
(bool)plcItem.GetAttribute("UseFixedSystemDiagnosticsAlarmIds");
// get value of "SuppressDeactivatingSystemDiagnosticsAlarms" via openness interface
bool isSuppressDeactivatingSystemDiagnosticsAlarms =
(bool)plcItem.GetAttribute("SuppressDeactivatingSystemDiagnosticsAlarms");
// set value of "UseFixedSystemDiagnosticsAlarmIds" via openness interface
plcItem.SetAttribute("UseFixedSystemDiagnosticsAlarmIds", false);
// set value of "SuppressDeactivatingSystemDiagnosticsAlarms" via openness interface
plcItem.SetAttribute("SuppressDeactivatingSystemDiagnosticsAlarms", true);
```

5.11.1.2 Downloading PC System

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to download PC-Station Plus and the SW-CPU independently. In order to download the PC-Station Plus configuration, you must retrieve stationmanager device item of rack of device and get the download provider as a service. SW-CPU itself must be retrieved as a device item for the same operation.

Downloading SW-CPU is similar to downloading a regular plc. For downloading SW-CPU, please see Downloading to PLC devices (Page 385).

Restriction

Download of F-Activated PLCs is not currently supported for SW-CPU as well; as it is in HW-CPU.

Program code: Download software and hardware components

You are able to download software and hardware components to a device via DownloadProvider service that can be acquired from a given DeviceItem. If a DeviceItem represents a downloadable target an instance of DownloadProvider will be returned on GetService call. Otherwise it will be null.

```
DeviceItem stationManager =  dev.DeviceItems.First(p => p.PositionNumber ==
0).DeviceItems.First(a => a.PositionNumber == 125);
DownloadProvider downloadProviderStationManager =
stationManager.GetService<DownloadProvider>();
if (downloadProviderStationManager == null)
{
  //no download is possible for PC-Station
}
DeviceItem swCpu = dev.DeviceItems.First(p => p.Name == "Software PLC_1");
DownloadProvider downloadProviderSwCpu = swCpu.GetService<DownloadProvider>();
```

Modify the following program code to configure Network parameters:

```
ConnectionConfiguration connConfig = downloadProviderStationManager.Configuration;
ConfigurationMode configurationMode = connConfig.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface =  configurationMode.PcInterfaces.Find("ASIX AX88179
USB 3.0 to Gigabit Ethernet Adapter", 1);
ConfigurationTargetInterface targetInterface = pcInterface.TargetInterfaces.Find("2 X2");
IConfiguration targetConfiguration = pcInterface.TargetInterfaces[0];
bool isConfigured = connConfig.ApplyConfiguration(targetInterface);
if (isConfigured)
...
```

It must be noted that, in case the target system reboots after downloading PC-Station Plus configuration, the default behavior in openness is **waiting**. This means that, the openness code flow is stopped until the first stage download is successful (and target system is rebooted) or timed out or failed.

In case you do not want to wait for the reboot, you can use a post download option which is called WaitOnReboot

```
//Post Download Configuration Delegate
 DownloadConfigurationDelegate postDownloadForPcStation = downloadConfiguration =>
 {
 WaitOnReboot waitOnReboot = downloadConfiguration as WaitOnReboot;
 if (waitOnReboot != null)
 {
 //In case user does not want to wait...
 waitOnReboot.CurrentSelection = WaitOnRebootSelections.NoAction;
 //In case user wants to wait... This is the default option anyway...
 //waitOnReboot.CurrentSelection = WaitOnRebootSelections.Wait;
 return;
 }
 };
```

You must continue downloading SW-CPU separately, but it must be made sure that the download of PC-Station was executed before.

```
DownloadResult downloadResult = null;
try
{
//WE FIRST DOWNLOAD PC-STATION
downloadResult = downloadProviderStationManager.Download(targetConfiguration,
preDownloadForPcStation, postDownloadForPcStation, DownloadOptions.Hardware);
if (DownloadResultState.Error != downloadResult.State)
{
Console.WriteLine("The download is successful for pc-station");
}
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine("Exception Thrown, Message: " + e.Message.ToString());
}
downloadResult = null;
try
{
downloadResult = downloadProviderSwCpu.Download(targetConfiguration,  preDownloadForSwCpu,
postDownloadForWinac, DownloadOptions.Hardware | DownloadOptions.Software);
if (DownloadResultState.Error != downloadResult.State)
{
Console.WriteLine("The download is successful for SW-CPU");
}
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine("Exception Thrown, Message: " + e.Message.ToString());
 }
```

Downloading SW-CPU is similar to downloading a regular plc. For downloading SW-CPU, please see Downloading to PLC devices (Page 385).

Note

Download of F-Activated PLCs is not currently supported for SW-CPU as well; as it is in HW-CPU.

A download method also triggers compilation. It is also possible to compile a device item stand alone, with the following code snippet:

```
ICompilable compileServiceSwCpu = swCpu.GetService<ICompilable>();
ICompilable compileServiceStationManager = stationManager.GetService<ICompilable>();
CompilerResult compileResultStationManager = compileServiceStationManager.Compile();
CompilerResult compileResultSwCpu = compileServiceStationManager.Compile();
bool compileCheck = !compileResultStationManager.State.Equals(CompilerResultState.Error);
if (compileCheck != true)
{
...
}
```

5.11.1.3 Downloading to PLC devices

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to download software and hardware components to PLC device through DownloadProvider (accessed from the DeviceItem). If a DeviceItem represents a downloadable target, an instance of DownloadProvider will be returned on GetService call, else the service will return null.

Program code: Retrieving DownloadProvider service from a device item

```
DeviceItem deviceItem = ...;
DownloadProvider downloadProvider = deviceItem.GetService<DownloadProvider>();
if (downloadProvider != null)
{
...
}
```

In order to execute a download to a device, you will have to call Download method of DownloadProvider.

The parameters are an IConfiguration object, two delegates and DownloadOptions (Hardware, Software or Hardware and Software).

```
...
IConfiguration targetConfiguration = ...
DownloadConfigurationDelegate preDownloadDelegate = ...
DownloadConfigurationDelegate postDownloadDelegate = ...
DownloadOptions loadOption = DownloadOptions.Hardware | DownloadOptions.Software;
...
DownloadResult result = downloadProvider.Download(targetConfiguration,
preDownloadDelegate, postDownloadDelegate, loadOption);
```

An optional parameter can be used for downloading to an IP address where the IP address of the running PLC differs from the IP address configured offline.

```
...
ConfigurationAddress onlineAddress = ...
DownloadResult result = downloadProvider.Download(targetConfiguration, onlineAddress,
preDownloadDelegate, postDownloadDelegate, loadOption);
```

Parameters of download method

In order to download to a PLC device, you have to call Download method of DownloadProvider. The Download method has four parameters which are IConfiguration object, two delegates and DownloadOptions (Hardware, Software or Hardware and Software).

Parameter name	Type	Description
configuration	Siemens.Engineering.Connection.IConfiguration	Connection configuration to a device.
onlineAddress	Siemens.Engineering.Connection.ConfigurationAddress	Optional parameter for download to a changed IP address
preDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate to be called for checking configuration before download operation.
postDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate to be called for checking configuration after download operation.
downloadOptions	Siemens.Engineering.Download.DownloadOptions	Download options.

- Openness download is supported only if the configurations that are listed in above table are encountered and properly handled by you. If the configuration is invalid, then EngineeringTargetInvocationException is thrown and download process is aborted.
- Since compliation is a part of download, it is recommended to compile before download to get a full feedback about the complile results.

IConfiguration

You will have to provide IConfiguration object to the Download method. It will be used to establish a connection to the given PLC device. IConfiguration interface is implemented by ConfigurationAddress and ConfigurationTargetInterface. Both the objects can be accessed through ConnectionConfiguration instance. ConnectionConfiguration instance can be acquired from DownloadProvider.Connection: ConnectionConfiguration or optionally from OnlineProvider.Connection: ConnectionConfiguration properties.

Configuration of ConnectionConfiguration object is described in Accessing parameters of an online connection (Page 434) section.

```
...
DownloadProvider downloadProvider = null;
ConnectionConfiguration configuration = downloadProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
IConfiguration targetConfiguration = pcInterface.TargetInterfaces[0];
...
```

ConfigurationAddress

If the projected IP-address differs from the already downloaded online IP-address, an additional parameter has to use. This is for identify the correct online PLC.

```
 ...
DownloadProvider downloadProvider = ...
ConnectionConfiguration configuration = downloadProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
IConfiguration targetConfiguration = pcInterface.TargetInterfaces[0];
// Create the address object to reach the PLC online.
ConfigurationTargetInterface targetConfigurationIF = targetConfiguration as
ConfigurationTargetInterface;
ConfigurationAddressComposition downloadAddresses = targetConfigurationIF.Addresses;
ConfigurationAddress onlineAddress = downloadAddresses.Create("10.119.57.223");
DownloadResult result = downloadProvider.Download(targetConfigurationIF, onlineAddress,
preDownloadDelegate, postDownloadDelegate, loadOption);
...
```

DownloadConfigurationDelegate

You will have to provide two implementations of void DownloadConfigurationDelegate(DownloadConfiguration downloadConfiguration). First delegate will be called for pre-download configurations, and the second will be called after download is completed. The delegates will be called for each configuration that requires an action from the user. For more information about callback handling, Refer Supporting callbacks (Page 406). . If configurations are not handled properly, an EngineeringTargetInvocationException will be thrown and download process will be aborted. .Certain configurations will only contain an information, therefore your action will not be required. Also, you can skip the configurations that do not prevent the download.

Note

You should avoid any modifications other than those required to resolve the download configurations in their delegate implementation.

The possible download configuration types are listed below.

Configuration name	Description and properties
DownloadConfiguration	• Base class for all the configurations. It will contain information only if it cannot be cast to a specific configuration. IConfiguration has a single property: • DownloadConfiguration.Message : string (read only property contains the configuration message)
DownloadSelectionConfiguration	• Base class for all configuration that can be selected. Properties of DownloadSelectionConfiguration • Does not contain additional properties. A selection must be provided in all child classes derived from it.

Configuration name	Description and properties
DownloadCheckConfiguration	• Base class for all configuration that can be checked and unchecked. • Contains single property DownloadCheckConfiguration.Checked: bool<string> (read/write property identifies whether the configuration is checked or unchecked)
DownloadPasswordConfiguration	• Base class for all configuration that required a password to proceed with download (PLC protection passwords and block binding passwords). Methods of DownloadPasswordConfiguration: • Conains a single method to set password. DownloadPasswordConfiguration.SetPassword (password: SecureString) : void - Sets password. SecureString

The datatype of the configurations are given below.

Configuration	DataType	Description and Action

DownloadSelection-Configuration	StartModules	`Set CurrentSelection:StartModulesSelections`. Available enum values are
		`NoAction (No action)`
		`StartModule (Stop module)`
		These modules are stopped for downloading to a device.

	StopModules	`Set CurrentSelection:StopModulesSelections`. Available enum values:
		`NoAction (No action)`
		`StopAll (Stop all)`
		These modules are started after the download operation.

	AllBlocksDownload	`Set CurrentSelection:AllBlocksDownloadSelections`. Available enum value is `DownloadAllBlocks (Download all blocks to the device)` Download software to device

	OverwriteSystemData	`Set CurrentSelection:OverwriteSystemDataSelections`. Available enum values are
		`NoAction (No action)`
		`Overwrite (Download to device)`
		Delete and replace the existing system data in target location.

	ConsistentBlocksDown-load	`Set CurrentSelection:ConsistentBlocksDownloadSelections`. Available enum value is `ConsistentDownload (Consistent download)` Download the software to device.
	AlarmTextLibrariesDown-load	`Set CurrentSelection:AlarmTextLibrariesDownloadSelections`. Available enum values are ConsistentDownload (Consistent download) NoAction (No action) Download all alarm texts and text list texts.

	ProtectionLevelChanged	Set CurrentSelection:ProtectionLevelChangedSelections. Available enum values are
		`NoChange (No change)`
		`ContinueDownloading (Continue downloading to the device)`
		CPU protection is changed to the next lower level.

	ActiveTestCanBeAborted	`Set CurrentSelection:ActiveTestCanBeAbortedSelections`. Available enum values are
		`NoAction (No action)`
		`AcceptAll (Accept all)`
		Active test and commissioning functions are canceled during the load-ing operation of the device.

	ResetModule	`Set CurrentSelection:ResetModuleSelections` Available enum values are

Configuration	DataType	Description and Action

		`NoAction (No action)`
		`DeleteAll (Delete all)`
		It resets the module.

	LoadIdentificationData	`Set CurrentSelection:LoadIdentificationDataSelections.` Available enum values are
		`LoadNothing (Load nothing)`
		`LoadData (Load data)`
		Load identification data to the PROFINET IO devices and their modules.

	DifferentTargetConfiguration	`Set CurrentSelection:DifferentTargetConfigurationSelections.` Available enum values are
		`NoAction (No action)`
		`AcceptAll (Accept all)`
		Gives the difference between configured and target modules (online)

	InitializeMemory	`Set CurrentSelection:InitializeMemorySelections.` Available enum values:
		`NoAction (No action)`
		`AcceptAll (Accept all)`
		This datatype is used to initialize memory.

	ExpandDownload	`Set CurrentSelection: ExpandDownloadSelections.` Available enum values:
		`NoAction (No action)`
		`Download (Download)}}`
		The download must be expanded beyond your selection.

	ActiveTestCanPrevent-Download	`Set/Get CurrentSelection:ActiveTestCanPreventDownloadSelections`
		Available enum values:
		`NoAction (No action)`
		`AcceptAll (Accept all)`
		Active test and commissioning functions could prevent the donwload.

	DataBlockReinitilization	`Set CurrentSelection:DataBlockReinitializationSelections`. Available enum values: `StopPlcAndReInitialize` (All data values, including retain data, will be initialized with their defined start values during loading. Set the PLC to STOP before loading.) `NoAction` (No action regarding "Download with reinitialization".) Different memory reserves in the online block and offline block are preventing "Download without reinitialization".

Configuration	DataType	Description and Action
DownloadCheckConfiguration	CheckBeforeDownload	`Set IsChecked:bool` property. Checks before downloading to the device.
	UpgradeTargetDevice	`Set IsChecked:bool` property. Checks the different project versions in the configured device and target device (online).
	OverWriteHMIData	`Set IsChecked:bool` property. Overwrites if object exists online.
	FitHMIComponents	`Set IsChecked:bool` property. Components with a different version are installed on the target device.
	TurnOffSequence	`Set IsChecked:bool` property Turns off the sequence before loading.
	OverwriteTargetLanguages	`Set IsChecked:bool` property. To distinguish the settings between the project and PLC programming
	DowngradeTargetDevice	`Set IsChecked:bool` property. To mention the different data formats in online and offline projects.
DownloadPassword-Configuration	ModuleReadAccessPassword	`Set password via SetPassword(password:SecureString)` method. Enter a password to gain read access to the module.
	ModuleWriteAccessPassword	`Set password via SetPassword(password:SecureString)` method. Enter a password to gain write access to the module.
	BlockBindingPassword	`Set password via SetPassword(password:SecureString)` method. Block binding password configuration method.
	PlcMasterSecretPassword	`Set password via SetPassword(password:SecureString)` method. Used for entering the PLC Master Secret password

Note

The configurations that enable to enter a password have an additional attribute

NOTICE

Please note that download configurations are similar to configurations encountered in Load preview and Load results dialogs while working with GUI of TIA Portal.

⚠ WARNING

The API user is responsible for ensuring the security measures of handling passwords through code.

IsSecureCommunication

The attribute provides 'TRUE' in the case of a "Secure communication" (TLS handshake) connection to the Plc. In all other cases the attribute delivers 'FALSE'. For more information on IsSecureCommunication, refer Supporting secure S7 communication TLS (Page 475)

So this concerns:

- ModuleReadAccessPassword
- ModuleWriteAccessPassword
- BlockBindingPassword
- PlcMasterSecretPassword

```
bool isSecureCommunication = moduleReadAccessPassword.IsSecureCommunication;
if (isSecureCommunication == true)
{
// Secure communication with (Tls handshake)
...
}
```

Unhandled exceptions inside the delegates

Unhandled configuration that can prevent the download causes an EngineeringTargetInvocationException and aborts download. An EngineeringDelegateInvocationException will be thrown in case of an unhandled exception within the Delegate.

You can use the below code example for PreDownloadDelegate implementation:

```
private static void PreConfigureDownload(DownloadConfiguration downloadConfiguration)
{
StopModules stopModules = downloadConfiguration as StopModules;
if (stopModules != null)
{
stopModules.CurrentSelection = StopModulesSelections.StopAll; // This selection will set
PLC into "Stop" mode
return;
}
AlarmTextLibrariesDownload alarmTextLibraries = downloadConfiguration as
AlarmTextLibrariesDownload;
if (alarmTextLibraries != null)
{
alarmTextLibraries.CurrentSelection =
AlarmTextLibrariesDownloadSelections.ConsistentDownload;
return;
}
BlockBindingPassword blockBindingPassword = downloadConfiguration as BlockBindingPassword;
if(blockBindingPassword != null)
{
SecureString password = ...;// Get Binding password from a secure location
blockBindingPassword.SetPassword(password);
return;
}
CheckBeforeDownload checkBeforeDownload = downloadConfiguration as CheckBeforeDownload;
if (checkBeforeDownload != null)
{
checkBeforeDownload.Checked = true;
return;
}
ConsistentBlocksDownload consistentBlocksDownload = downloadConfiguration as
ConsistentBlocksDownload;
if (consistentBlocksDownload != null)
{
consistentBlocksDownload.CurrentSelection =
ConsistentBlocksDownloadSelections.ConsistentDownload;
return;
}
ModuleWriteAccessPassword moduleWriteAccessPassword = downloadConfiguration as
ModuleWriteAccessPassword;
if (moduleWriteAccessPassword != null)
{
SecureString password = ...;// Get PLC protection level password from a secure location
moduleWriteAccessPassword.SetPassword(password);
return;
}
throw new NotSupportedException(); // Exception thrown in the delagate will cancel download
}
```

You can use the below code example for PostDownloadDelegate implementation:

```
private static void PostConfigureDownload(DownloadConfiguration downloadConfiguration)
{
StartModules startModules = downloadConfiguration as StartModules;
if (startModules != null)
{
startModules.CurrentSelection = StartModulesSelections.StartModule; // Sets PLC in "Run"
mode
}
}
```

DownloadOptions

You must specify the download options through DownloadOptions flagged enum. This parameter will determine the type of download to be performed such as Hardware, Software, SoftwareOnlyChanges or a combination of them.

```
[Flags]
public enum DownloadOptions
{
None = 0, // Download nothing
Hardware = 1, // Download hardware. Complete download when used alone, delta download when
used in combination with Software or SoftwareOnlyChanges.
Software = 2, // Download software all. Do not combine with SoftwareOnlyChanges
SoftwareOnlyChanges = 4 // Download software only changes. Do not combine with Software
}
```

Possible DownloadOptions combinations (C# code)	System Reactions
none	exception
Hardware	HW complete
Software	SW complete
Hardware | Software	HW delta, SW complete
SoftwareOnlyChanges	SW delta
Hardware | SoftwareOnlyChanges	HW delta, SW delta
Software | SoftwareOnlyChanges	exception
Hardware | Software | SoftwareOnlyChanges	exception
(DownloadOptions)11	exception

DownloadResult

The DownloadResult returned by the Download action provides feedback on the state of the objects that were downloaded.

You can use the below code example for Download invocation:

```
[STAThread]
static void Main()
{
...
DownloadProvider downloadProvider = ...;
IConfiguration targetConfiguration = ...;
DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
DownloadResult result = downloadProvider.Download(targetConfiguration,
preDownloadDelegate, postDownloadDelegate, DownloadOptions.Hardware |
DownloadOptions.Software);
if (result.State == DownloadResultState.Error)
{
// Handle error state
}
WriteDownloadResults(result);
...
}
private static void PreConfigureDownload(DownloadConfiguration downloadConfiguration)
{
...
}
private static void PostConfigureDownload(DownloadConfiguration downloadConfiguration)
{
...
}
private void WriteDownloadResults(DownloadResult result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);
RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(DownloadResultMessageComposition messages, string
indent = "")
{
indent += "\t";
foreach (DownloadResultMessage message in messages)
{
Console.WriteLine(indent + "DateTime: " + message.DateTime);
Console.WriteLine(indent + "State: " + message.State);
Console.WriteLine(indent + "Message: " + message.Message);
RecursivelyWriteMessages(message.Messages, indent);
}
}
```

5.11.1.4 Downloading PLC to a Windows folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to download a PLC to a Windows Folder on the Windows file system. It is possible to automate the process of creating the SIMATIC memory card, along with the PLC program, without requiring a connection to the PLC or PLCSIM Advanced.

The feature is valid for the following products / devices:

- S7-1500 CPU including Safety (FW >= 2.1)
- S7-1500R/H CPU including Safety (HF)
- ET 200SP CPU including Safety
- ET 200pro CPU including Safety
- S7-1500 Software Controller including Safety
- ET 200SP Open Controller including Safety
- S7-1200 CPU including Safety (FW >= V4.2)

Parameter

The DownloadProvider.Download() and RHDownloadProvider.Download() accepts the following definition of parameter. Both the methods return a DownloadResult.

Parameter name	Type	Description
directoryInfo	System.IO.DirectoryInfo	Specifies the path of windows folder to save the configuration
preDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Specifies the delegate that will be called to check configuration before download.

For more information about details of DownloadConfigurationDelegate and DownloadResult, see Downloading to PLC devices (Page 385)

The following configurations are supported to download PLC to Windows Folder:

Data type	Action
OverwriteOnMemoryCard	Set CurrentSelection: OverwriteOnMemoryCardSelections Available enum values: • NoAction (No action) • Load (Download to device)
TargetForSoftware	Set CurrentSelection: TargetForSoftwareSelections Available enum values: • CPU (CPU) • PlcSimulationAdvanced (PLC Simulation Advanced)
ConsistentBlocksDownload	Set CurrentSelection: ConsistentBlocksDownloadSelections Available enum values: ConsistentDownload (Consistent download)
AlarmTextLibrariesDownload	Set CurrentSelection: AlarmTextLibrariesDownloadSelections Available enum values: • ConsistentDownload (Consistent download) • NoAction (No action)
SafetyProgram	Set CurrentSelection: SafetyProgramSelections Available enum values: ConsistentDownload (Consistent download)

For more information about other download configuration types supported, see Downloading to PLC devices (Page 385)

Program code: Download To Windows Folder Specifics

Modify the following program code to access DownloadProvider for standard Plc:

```
DeviceItem deviceItem = ...;
DownloadProvider downloadProvider = deviceItem.GetService<DownloadProvider>();
downloadProvider.Download(new DirectoryInfo(@"c:\tmp\card1"),
DownloadConfigurationDelegate);
```

Note

You must specify the local Windows folder on the Windows file system and a Delegate method.

Modify the following program code to access RHDownloadProvider for R/H system:

```
Device device = ...;
RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
rhDownloadProvider.Download(new DirectoryInfo(@"c:\tmp\card1"),
DownloadConfigurationDelegate);
```

Note

Although R/H systems are composed of two PLCs, only a single Download-Action is necessary. The saved configuration will be applied to update both the primary and backup PLC simultaneously.

For a successful download execution, the Openness program must manage the following configurations

```
private static void DownloadConfigurationDelegate(DownloadConfiguration
downloadConfiguration)
{
if (downloadConfiguration is TargetForSoftware configTargetForSoftware)
{
configTargetForSoftware.CurrentSelection =TargetForSoftwareSelections.CPU;
}
else if (downloadConfiguration is OverwriteOnMemoryCard configOverwriteOnMemoryCard)
{
configOverwriteOnMemoryCard.CurrentSelection = OverwriteOnMemoryCardSelections.Load;
}
else if (downloadConfiguration is ConsistentBlocksDownload configConsistentBlocksDownload )
{
configConsistentBlocksDownload.CurrentSelection =
ConsistentBlocksDownloadSelections.ConsistentDownload;
}
else if (downloadConfiguration is AlarmTextLibrariesDownload
configAlarmTextLibrariesDownload)
{
configAlarmTextLibrariesDownload.CurrentSelection =
AlarmTextLibrariesDownloadSelections.ConsistentDownload;
}
else if (downloadConfiguration is SafetyProgram safetyProgram)
{
safetyProgram.CurrentSelection = SafetyProgramSelections.ConsistentDownload;
}
}
```

Program code: DownloadResult Example

```
String directory = ...;
var downloadResult = downloadProvider.Download(new DirectoryInfo(directory),
DownloadConfigurationDelegate);
PrintAllMessages(downloadResult.Messages, 0, downloadResult);
```

5.11.1.5 Downloading Master Secret to PLC

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to download Master Secrets to PLCs through PlcMasterSecretPassword accessed from download configuration. To download, you will have to provide an implementation of void DownloadConfigurationDelegate (DownloadConfiguration downloadConfiguration) for the pre-download configuration.

This delegate will be called for each configuration that requires an action from the user. The PlcMasterSecretPassword configuration handles the user action for entering the Master Secret password.

For more information about callback handling and Support of download to PLC devices, Refer Supporting callbacks (Page 406) and Downloading to PLC devices (Page 385)

Program code: PlcMasterSecretPassword Donwload

You can use the below code sample to download software and hardware components to a device via DownloadProvider service. For information on IsSecureCommunication, refer Supporting secure S7 communication TLS (Page 475)

```
internal bool DownloadDevice(DeviceItem deviceItem)
{
try
{
//Download software and hardware components to a device via DownloadProvider service that
//can be acquired from a given DeviceItem. If a DeviceItem represents a download able
//target an instance of DownloadProvider will be returned on GetService call. Otherwise
//it will be null.
DownloadProvider myDownloadProvider = deviceItem.GetService<DownloadProvider>();
if (myDownloadProvider == null)
{
return false;
}
//Preparation for IConfigurable object to establish a connection to the device.
IConfiguration
//interface is implemented by ConfigurationAddress and ConfigurationTargetInterface. Both
//objects can be accessed via ConnectionConfiguration instance. ConnectionConfiguration
//instance can be acquired from DownloadProvider.Configuration
ConnectionConfiguration connConfiguration = myDownloadProvider.Configuration;
ConfigurationMode confMode = connConfiguration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = ("Intel(R) Ethernet Connection I217-LM", 1);
ConsoleOutput.PrintInterfaces(connConfiguration);
//In order to execute a download to a device, Download method of DownloadProvider have to
//be called. The parameters are an IConfiguration object, two delegates and DownloadOptions
// (Hardware, Software or Hardware and Software).
IConfiguration targetConfiguration = pcInterface.TargetInterfaces[0];
DownloadConfigurationDelegate preDownloadConfigurationDelegate = PreConfigureDownload;
DownloadConfigurationDelegate postDownloadConfigurationDelegate = PostConfigureDownload;
DownloadOptions downloadOptions = DownloadOptions.Hardware | DownloadOptions.Software;
// Apply the the configuration and start the download
myDownloadProvider.Configuration.ApplyConfiguration(pcInterface.TargetInterfaces.First());
var downLoadResult = myDownloadProvider.Download(targetConfiguration,
preDownloadConfigurationDelegate,
postDownloadConfigurationDelegate, downloadOptions);
return downLoadResult != null && downLoadResult.State != DownloadResultState.Error;
}
catch (EngineeringTargetInvocationException ex)
{
//Exception catch-ed on purpose. Client can write own code to handle the exception
//In case of PLC Master Secret exception will be thrown if:
// 1) If a different Master Secret has already been set in the PLC.
//Exception Message: An error has occurred during download: 'A different Master Secret
//has already been set. The PLC has to be reset!
//Possibly provide custom exception handling here
Console.WriteLine("An exception during download was thrown:");
Console.WriteLine(ex.DetailMessageData);
return false;
}
}
```

You can use the below code sample for PlcMasterSecretPassword handling in PreDownloadDelegate:

```
private static void PreConfigureDownload(DownloadConfiguration downloadConfiguration)
{... ... ... ... ...
// Set the PLC Master Secret password configured in the offline project's PLC,
// when the password was never set in the real PLC
if (downloadConfiguration is PlcMasterSecretPassword plcMasterSecret)
{
// Get the PLC Master Secret password from a secure location
SecureString plcMasterSecretPsw = ...;
plcMasterSecret.SetPassword(plcMasterSecretPsw);
return;
}
catch (EngineeringTargetInvocationException ex)
{
// Exception catch-ed on purpose. Client can write own code to handle the exception
// Exception will be thrown if:
// 1) plcMasterSecret password is null.
// Exception Message: The argument "password" may not be null
// 2) plcMasterSecret password is an empty string.
// Exception Message: The password provided is invalid
// 3) plcMasterSecret password is wrong.
// Exception Message: The password provided is invalid
// Possibly provide custom exception handling here
Console.WriteLine("PlcMasterSecretPassword has thrown an exception:");
Console.WriteLine(ex.DetailMessageData);
return;
}
}
... ... ... ... ...
throw new NotSupportedException();
// Exception thrown in the delagate will cancel download
}
```

Note

An EngineeringDelegateInvocationException will be thrown in case of an unhandled exception within the Delegate.

5.11.1.6 Managing PLC Master Secret in PLCs

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to protect the sensitive PLC configuration data. You can configure the PLC master secret key which will be used to encrypt the certificate configuration.

Additionally, you can disable if protecting the PLC configuration data is not a priority.

You can also use the TIA Portal Openness to configure the PLC master secret key via the HW feature PlcMasterSecretConfigurator at the device item CPU of device. The feature provides actions to protect, unprotect and change the master secret key.

- void Protect(SecureString password)
- void Unprotect(SecureString password)
- void Unprotect()
- void ChangePassword(SecureString oldPassword, SecureString newPassword)
- attribute MasterSecretConfiguration MasterSecretConfiguration{get;}
- void Reset()

```
enum MasterSecretConfiguration
{
None = 0 , // The checkbox option “Protect the PLC Configuration data…” is unchecked.
WithPassword = 1, // The PLC Master Secret is configured.
WithoutPassword = 2 // The checkbox option “Protect the PLC Configuration data…” is
//checked and the PLC Master Secret is not configured.
}
```

Configuring PLC master secret key

Configuring PLC master secret key is implemented as SecureString for security reasons. It is only possible to set / change the passwords . Reading the passwords is not supported.

```
PlcMasterSecretConfigurator plcMasterSecretConfigurator =
plcCpu.GetService<PlcMasterSecretConfigurator>();
plcMasterSecretConfigurator.Protect(StringToSecureString(“TiaPassword123”));
plcMasterSecretConfigurator.ChangePassword(StringToSecureString(“TiaPassword123”),
StringToSecureString(“TiaPassword1234”));
public SecureString StringToSecureString(string value)
{
SecureString secureStr = new SecureString();
if (!string.IsNullOrEmpty(value))
{
for(int i = 0; i < value.Length; i++)
{
secureStr.AppendChar(value[i]);
}
}
return secureStr;
}
```

Disabling PLC Master Secret Functionality

Disabling PLC Master Secret password is implemented using overloaded Unprotect methods , based on the master secret configuration you can disable PLC master secret functionality with and

without master secret key. Reading the passwords is not supported.

```
//Unprotect method that will be called when PlcMasterSecret is not configured
plcMasterSecretConfigurator.Unprotect();

//Unprotect method that will be called when PlcMasterSecret is configured
plcMasterSecretConfigurator.Unprotect(StringToSecureString("TiaPassword123"));
```

Reset PLC Master Secret Functionality

If you forget master secret key , the reset will remove the Master secret configuration and the certificates will be lost.

```
plcMasterSecretConfigurator.Protect(StringToSecureString("TiaPassword123"));
plcMasterSecretConfigurator.Reset();
```

5.11.1.7 Setting / deleting a PLC Master Secret in a PLCs

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to set or reset the PlcMasterSecret to PLCs. To achieve the functionalities in the TIA Portal Openness,-OnlineProvider are extended by new Methods:

- SetPlcMasterSecret()
- ResetPlcMasterSecret()

The SetPlcMasterSecret() supports the following parameter:

Parameter	Type	Description
securePassword	SecureString	The Master Secret-password.

Note

You cannot delete or reset the Plc Master Secret by a NULL-password.

Program code: Set and Reset PlcMasterSecret

Modify the following program code to set and reset the PlcMasterSecret via the OnlineProvider service:

```
DeviceItem deviceItem = ...;
SecureString password = ...;
OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
```

Modify the following program code to set the PlcMasterSecret:

```
DeviceItem deviceItem = ...;
SecureString password = ...;
OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
onlineProvider.SetPlcMasterSecret(password);
```

Modify the following program code to reset the PlcMasterSecret:

```
DeviceItem deviceItem = ...;
OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
onlineProvider.ResetPlcMasterSecret();
```

Note

Reset the Plc Master Secret is the same functionality like Delete in the "Online & diagnostics"-Dialog.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.1.8 Running and stopping PLC

Requirements

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

When interacting with TIA Portal through Openness API, it may be necessary to change the operating mode of the PLC. TIA Portal Openness provides a way to modify the operating state of the PLC either to start or stop.

Program code

Modify the following program code for setting PLC operating state to STOP.

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
 {
    StopModules stopModules = downloadConfiguration as StopModules;
    if (stopModules != null)
         {
         // Puts PLC in "Stop" mode
         stopModules.CurrentSelection = StopModulesSelections.StopAll;
    }
}
```

Modify the following program code for setting PLC operating state to START.

```
public void ConfigurePostDownload(DownloadConfiguration downloadConfiguration)
  {
   StartModules startModules = downloadConfiguration as StartModules;
      if (startModules != null)
         {
          // Puts PLC in "Start" mode
          startModules.CurrentSelection = StartModulesSelections.StartModule;
    }
}
```

5.11.1.9 Supporting callbacks

Requirements

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Certain API methods require an interaction with the user-defined application code during their execution. Delegates are used to handle these callback actions in the user-defined application code. You need to implement a method with a compatible signature, and pass it as a delegate parameter to the action. To proceed with the execution, TIA Portal calls the implemented methods.

Program code

```
// This delegate is declared in Siemens.Engineering.dll
public delegate void
Siemens.Engineering.Download.DownloadConfigurationDelegate(Siemens.Engineering.Download.Co
nfigurations.DownloadConfiguration configuration);
...
```

Example of an user application code using and implementing the delegate:

```
[STAThread]
static void Main()
{
  ...
    DownloadProvider downloadProvider = ...;
    IConfiguration targetConfiguration = ...;
    DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
    DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
    DownloadResult result = downloadProvider.Download(targetConfiguration,
preDownloadDelegate, postDownloadDelegate, DownloadOptions.Hardware |
DownloadOptions.Software);
  ...

}

//This method will be called back by TIA Portal
private static void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
   {
     // Work with the parameter
   }

//This method will be called back by TIA Portal
private static void ConfigurePostDownload(DownloadConfiguration downloadConfiguration)
    {
      // Work with the parameter
    }
```

Note

STAThread attribute will assure that the delegates are called in the Main thread of execution.

5.11.1.10 Protecting PLC through password

Requirements

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

When interacting with TIA Portal through Openness API, it may be necessary to change the protection level of the PLC. TIA Portal Openness provides a way to secure the PLC through a password. The password can be set to both read-protected and write-protected PLCs.

Program code

Modify the following program code for read-protected PLCs

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
  {
       ModuleReadAccessPassword moduleReadAccessPassword = downloadConfiguration
asModuleReadAccessPassword;
       if (moduleReadAccessPassword != null)
          {
            SecureString password = ...; // Get password from a secure location
            moduleReadAccessPassword.SetPassword(password); // enter the password to gain
full access
          }
  }
```

Modify the following program code for write-protected PLCs

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
  {
    ModuleWriteAccessPassword moduleWriteAccessPassword = downloadConfigurationas
ModuleWriteAccessPassword;
    if (moduleWriteAccessPassword != null)
     {
       SecureString password = ...; // Get password from a secure location
      moduleWriteAccessPassword.SetPassword(password); // enter the password to gain full
access
     }
  }
```

WARNING

The API user is responsible for ensuring the security measures of handling passwords through code.

5.11.1.11 Handling PLC block binding passwords

Requirements

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

TIA Portal Openness supports the data binding of passwords for customer applications. TIA Portal Openness provides a way for the customer to specify a block binding password. For example, a block binding password can be configured on the DownloadPasswordConfiguration class by calling the SetPassword method.

Note

If you want to secure the download action with a password, a password will have to be provided during every call of download function. This is regardless of whether the device has already been configured. After the successful acceptance of password for a given configuration, all subsequent calls to SetPassword are ignored.

Program code

Modify the following program code:

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    DownloadPasswordConfiguration downloadPasswordConfiguration = downloadConfiguration as
DownloadPasswordConfiguration;
    if(downloadPasswordConfiguration != null &&
downloadPasswordConfiguration.Message.Contains("block_1"))
       {
              SecureString password = ...; // Get password from a secured location
              downloadPasswordConfiguration.SetPassword(password);
    }
}
```

5.11.1.12 Uploading PLC device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to upload station into a project through StationUploadProvider. The StationUploadProvider instance is returned on GetService() invoked with a project to execute an upload.

The upload station is not supported for device group.

Program code: Retrieving StationUploadProvider service from a project

```
Project myProject = ...;
StationUploadProvider uploadProviderForProject =
myProject.GetService<StationUploadProvider>();
if (uploadProviderForProject != null)
{
    ...
}
```

Parameters of upload method

In order to execute an upload of a PLC device, you can call StationUpload() of StationUploadProvider. The Upload method has ConfigurationAddress and UploadConfigurationDelegte parameters.

UploadOptions are optional, because the StationUpload uploads Software, Hardware, and Files.

Parameter name	Type	Description
configurationAddress	Siemens.Engineering.Connection.ConfigurationAddress	Address of device that should be uploaded
uploadConfigurationDelegate	Siemens.Engineering.Upload.UploadConfigurationDelegate	Delegate that will be called to check configuration before upload
uploadOptions	Siemens.Engineering.Upload.UploadOptions	Upload options

Parameter 1: ConfigurationAddress

You should provide ConfigurationAddress object to the Upload. The address object is used to establish a connection to the given PLC device that should be uploaded. The ConfigurationAddress object must be created in the ConnectionConfiguration of the StationUploadProvider.

The Configuration contains a list of supported Modes. You need to select one of the Modes that should be used for upload. The selected ConfigurationMode contains a list of all local PcInterfaces that support the selected Mode, you have to select one of the interfaces. The desired address can be created in the Address collection of the selected ConfigurationPcInterface.

Modify the following code to create an address object:

```
...
StationUploadProvider uploadProvider = null;
...
ConnectionConfiguration configuration = uploadProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
//"Create an address. This "ConfigurationAddress" is used as parameter for upload."
ConfigurationAddress uploadAddress = pcInterface.Addresses.Create("192.68.0.1");
...
```

Project upload

You can start the station upload by calling the action StationUpload.

The following Parameters are mandatory:

- ConfigurationAddress: The address of the device to be uploaded
- UploadConfigurationDelegate: The callback to handle upload inhibits

```
...
StationUploadProvider uploadProvider = null;
Device uploadedObject = null;
...
UploadConfigurationDelegate preUploadDelegate = PreConfigureUpload;
UploadResult result = uploadProvider.StationUpload(uploadAddress, preUploadDelegate);
 // The uploaded device
uploadedObject = result.UploadedStation;
if (uploadedObject == null)
{
 ...
}
internal void PreConfigureUpload(UploadConfiguration uploadConfiguration)
{
 ...
}
```

Parameter2: UploadConfigurationDelegate

You need to provide an implementation of void UploadConfigurationDelegate (UploadConfiguration uploadConfiguration). The delegate will be called for pre-upload configurations. The delegate will be called for each configuration that requires an action from the user. For more information about callback handling, Refer Supporting callbacks (Page 406). Certain configurations will only contain an information, therefore the user action will not be required.

The possible upload configuration types are listed below:

Configuration name	Description and properties
UploadConfiguration	• Base class for all the configurations. It contains information in the Message attribute • Contains single property UploadConfiguration.Message : string (read only property contains the configuration message)
UploadPasswordConfiguration	• Derived from UploadConfiguration • Base class for all configuration that required a password for upload. • Contains a single method to set password. UploadPasswordConfiguration.SetPassword (password: SecureString) : void - Set password
UploadSelectionConfiguration	• Derived class of UploadConfiguration • Does not contain additional properties

The datatype of the configurations are given below:

Configuration	DataType	Description and Action
UploadPasswordConfiguration	ModuleReadAccessPassword	`Set password via SetPassword(password:SecureString)` method. Enter a password to gain read access to the module.
	PasswordReadAccess	`Set password via SetPassword(password:SecureString)` method. Enter a password for SW Upload in classic PLC's to gain read access to the module.

UploadSelectionConfiguration	UploadMissingProducts	`Set CurrentSelection:UploadMissingProductsSelections`
		Available enum values:
		`TryUpload (Consistent upload)`
		`NoAction (No action)`
		Set a selection for upload.

The support of a Failsafe password is not necessary. For the read-access by uploading a F-PLC no password is needed.

IsSecureCommunication

All configurations have the attribute: IsSecureCommunication The attribute provides 'TRUE' in the case of a "Secure communication" (TLS handshake) connection to the Plc. In all other cases the attribute delivers 'FALSE'. For more information on IsSecureCommunication, refer Supporting secure S7 communication TLS (Page 475)

```
bool isSecureCommunication = moduleReadAccessPassword.IsSecureCommunication;
if (isSecureCommunication == true)
{
// Secure communication with (Tls handshake)
...
}
```

Unhandled configuration that can prevent the upload causes an EngineeringTargetInvocationException and aborts upload.

An EngineeringDelegateInvocationException will be thrown in case of an unhandled exception within the Delegate.

PreUploadDelegate implementation example:

```
private static void PreConfigureUpload(UploadConfiguration UploadConfiguration)
{
ModuleReadAccessPassword moduleReadAccessPassword = UploadConfiguration as
ModuleReadAccessPassword;
if (moduleReadAccessPassword != null)
{
string passWD = "passWD";
var password = new SecureString();
foreach (var c in passWD)
password.AppendChar(c);
moduleReadAccessPassword.SetPassword(password);
return;
}
ModuleWriteAccessPassword moduleWriteAccessPassword = UploadConfiguration as
ModuleWriteAccessPassword;
if (moduleWriteAccessPassword != null)
{
string passWD = "passWD";
var password = new SecureString();
foreach (var c in passWD)
password.AppendChar(c);
moduleWriteAccessPassword.SetPassword(password);
return;
}
...
throw new NotSupportedException(); // Exception thrown in the delagate will cancel upload
}
```

Parameter3: UploadOptions

You cannot specify the Upload options. This Upload options are known as: "Hardware", "Software", "Hardware and Software" and "Hardware, Software and Files".

The StationUpload works automatically with the Upload options "Hardware, Software and Files".

Upload over Router

The TIA Portal Openness supports the stationupload via Router because the station behind a router in not visibly in the project navigator. For an upload over NAT-Router this address has configured with a mapped address.

Also an upload by using a Router with activated IP-Routing is possibly.

Upload over Router with IP-Routing

The device in the network behind the router can only be reached if the router supports IP-Routing or DNAT-Routing. In difference to a NAT-Router, the upload of the PLC is also possible if the PLC is not directly accessible but via a CP or a CM.

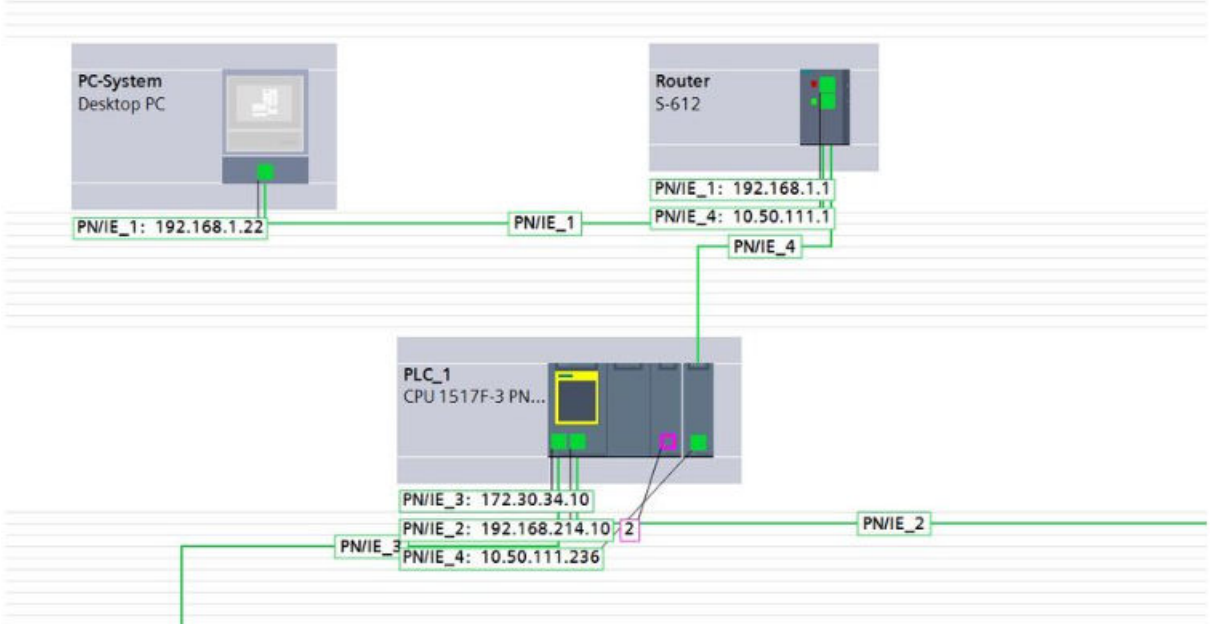
Therefore the following scenarios are supported:

- Connection between PG/PC and the PlusPLC that is directly connected to the destination subnet in combination with a router with IP-Routing.
- Connection between PG/PC and the PlusPLC that is directly connected to the destination subnet in combination with a router with DNAT-Routing
- Connection between PG/PC and PlusPLC via a communication module (CM) in combination with a router with IP-Routing
- Connection between PG/PC and PlusPLC via a communication module (CM) in combination with a router with DNAT-Routing
- Connection between PG/PC and PlusPLC via a communication processor (CP) in combination with a router with IP-Routing
- Connection between PG/PC and PlusPLC via a communication processor (CP) in combination with a router with DNAT-Routing

Note

The upload of a PLC-1200 V1 is not supported via a CM or CM; Only OMS+ protocol is supported.

For example, There is a device in a privat subnet. In this subnet the device has the address 192.168.214.10. Your PG/PC is in another subnet and has the address 192.168.1.22. Between these subnets you have a Router with IP-Routing. Now you can upload the device by using the address 10.50.111.236 of the CP.

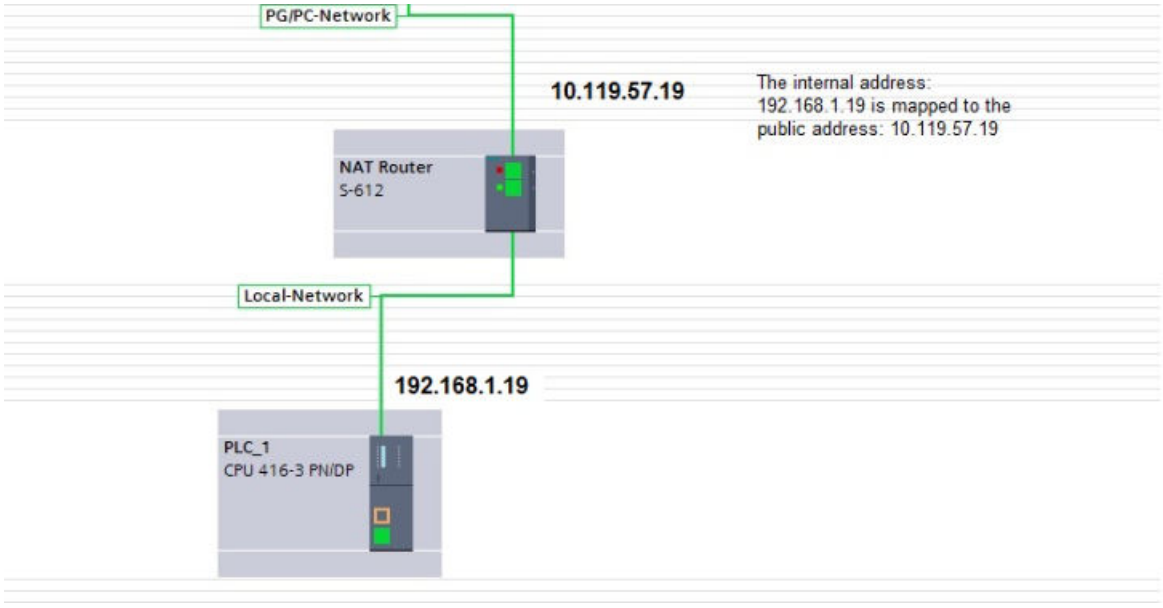


Upload over NAT-Router

A device in a private network can only be reached over a NAT Router, when the device IP-address is directly mapped to a public IP-address (destination NAT). Especially NAPT (network address and port translation) is not supported.

For example: There is a device in a privat subnet. In this subnet the device has the address 192.168.1.19. Your PG/PC is in another subnet and has the address 10.119.57.153. Between these subnets you have a NAT Router. The NAT Router needs a configuration, where the device address 192.168.1.19 is mapped to a valid address in the same subnet like the PG/PC. For example 10.119.57.19.

Now you can upload the device by using the address 10.119.57.19. (see section "**Configuration Address**").



There exist a few constrains for upload over NAT-Router.

At the first the Communication over a NAT router and S7-Routing cannot be used together. The device must be reachable directly by its IP-address from the NAT router. No S7-Router may be involved

in communication between NAT-router and device. That is the reason why the devices S71500 internal and S7300 internal cannot be reached from the PG/PC system.

The engineering system must be able to reach the NAT router directly by its IP-address. No S7-Router may be involved in communication between engineering system and NAT-router. That is the

reason while the engineering system PG/PC 2 in the above picture cannot be used for uploading a device from the Test Network.

UploadResult

The UploadResult returned by the Upload action provides feedback on the state of the objects that were uploaded.

- UploadResult.Message: UploadResultMessageComposition - Composition of UploadResultMessage

The following attributes are supported:

Attributes	Description
ErrorCount	int value of errors while upConnection between PG/PC and PlusPLC via a communication module (CM) in combination with a router with DNAT-Routingload
State	UploadResultState with possibly values: Success, Information, Warning and Error
UploadedStation	A Device-Instance of the uploaded station
WarningCount	Number of warning while upload as int

The UploadResultMessage contains:

- UploadResultMessage.Messages : UploadResultMessageComposition - Composition of UploadResultMessage

The following attributes are supported:

Attributes	Description
DateTime	System.DateTime of the created message.
ErrorCount	An int counter for errors.
State	UploadResultState with possibly values: Success, Information, Warning and Error.
WarningCount	Number of warning while upload as int

Upload invocation example

```
internal bool UploadPLC()
{
...
UploadResult result = uploadProvider.StationUpload(uploadAddress, preUploadDelegate);
...
PrintAllMessages(result.Messages, 0);
...
}
internal void PrintAllMessages(UploadResultMessageComposition messages, int level)
{
if (messages == null)
return;

if (level == 0)
Console.WriteLine("\n");
foreach (UploadResultMessage message in messages)
{
string messageOut = message.Message.PadLeft(message.Message.Length + level, '\t') + "\n";
Console.WriteLine(messageOut);
if ((message.Messages != null) && (message.Messages.Count > 0))
PrintAllMessages(message.Messages, level+1);
}
}
```

5.11.1.13 Comparing PLC software

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You have the following options to determine the deviation between the software of two devices:

- Comparing the software of two configured PLCs
- Comparison of the software of a PLC and the project library
- Comparison of the software of a PLC and the global library
- Comparison of the software of a PLC and the master copy of a PLC
- Comparisonof the software of a configured PLC with the software of a connected PLC in "Online" status

Signature

Use the `CompareTo` or `CompareToOnline` methods for the comparison.

```
public CompareResult CompareTo (ISoftwareCompareTarget
compareTarget)
```

```
public CompareResult CompareToOnline ()
```

Return value / parameter	Function
`CompareResult compareResult`	Returns the comparison result: • `FolderContentsDifferent`: Content of the compared folders differs. • `FolderContentsIdentical`: Content of the compared folders is identical. • `ObjectsDifferent`: Content of the compared objects differs. • `ObjectsIdentical`: Content of the compared objects is identical. • `LeftMissing`: The object is not contained in the object from which the comparison was started. • `RightMissing`: The object is not contained in the object which is being compared. • `CompareIrrelevant`: Comparison of these 2 objects is irrelevant • `FolderContainsDifferencesOwnStateDifferent`: Folder contents have one or more differences, folder's own state different • `FolderContentEqualOwnStateDifferent`: Folder content is the same, folder's own state is different
`ISoftwareCompareTarget compareTarget`	List of comparable objects.

Program code

Modify the following program code to output the comparison result:

```
private static void WriteResult(CompareResultElement compareResultElement, string indent)
{
    Console.WriteLine("{0} <{1}> <{2}> <{3}> <{4}> ",
    indent,
    compareResultElement.LeftName,
    compareResultElement.ComparisonResult,
    compareResultElement.RightName,
    compareResultElement.DetailedInformation);
    WriteResult(compareResultElement.Elements, indent);
}
private static void WriteResult (IEnumerable<CompareResultElement> compareResultElements,
string indent)
{
    indent += " ";
    foreach (CompareResultElement compareResultElement in compareResultElements)
    {
        WriteResult(compareResultElement, indent);
    }
}
```

Modify the following program code to compare the software of devices:

```
private static void CompareTwoOfflinePlcs(PlcSoftware plcSoftware0, PlcSoftware
plcSoftware1)
{
    if (plcSoftware0 != null && plcSoftware1 != null)
    {
        CompareResult compareResult = plcSoftware0.CompareTo(plcSoftware1);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Modify the following program code to compare the software of a PLC with the project library:

```
private static void ComparePlcToProjectLibrary(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(project.ProjectLibrary);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Modify the following program code to compare the software of a PLC with the global library:

```
private static void ComparePlcToGlobalLibrary(PlcSoftware plcSoftware, GlobalLibrary
globalLibrary)
{
    if (plcSoftware != null && globalLibrary != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(globalLibrary);
        WriteResult(compareResult.RootElement, String.Empty);
    }
}
```

Modify the following program code to compare the software of a PLC with a master copy:

```
private static void ComparePlcToMasterCopy(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult =
plcSoftware.CompareTo(project.ProjectLibrary.MasterCopyFolder.MasterCopies[0]);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Modify the following program code to compare the software of a PLC with the software of a connected PLC:

```
private static void ComparePlcToOnlinePlc(PlcSoftware plcSoftware)
{
    if (plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareToOnline();
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

5.11.1.14 Updating PLC program

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the UpdateProgram() to upgrade the instruction to the latest version of PLC. You can access the UpdateProgram() through PlcSoftware instance.

Program code

```
private void UpdateProgramPLC()
{
PlcSoftware plcSoftware = ...;
plcSoftware.UpdateProgram();
}
```

See also

Opening a project (Page 128)

5.11.1.15 Comparing PLC hardware

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to compare the hardware of two PLC devices.

Signature

Use the CompareTo method for the comparison of two hardware objects.

CompareResult CompareTo (IHardwareCompareTarget compareTarget);

Return value/parameter	Function
CompareResult compare result	Return the comparision result: • FolderContainsDifferencesOwnStateDifferent: Folder contents have one or more differences, folder's own state is different • FolderContentEqualOwnStateDifferent: Folder content is the same, folder's own state is different.
IHardwareCompareTarget compareTarget	The compare target for which the hardware compare should be performed. Must not be null.

If the Parameter compareTarget is null and an attempt is made to compare the hardware will throw Siemens.Enginnering.EngineeringTargetInvocationExceptions.

Program

Modify the following program code to output the comparision result:

```
...
CompareResult compareResult = plc_1.CompareTo(plc_2);
CompareResultState resultState = compareResult.RootElement.ComparisonResult;
if (resultState == CompareResultState.FolderContainsDifferencesOwnStateDifferent)
{
 // Folder contents have one or more differences, folder's own state is different:
 // May occur if the plc has a different subordinate element, e.g., a local module, and
 // the plc itself is different, e.g., in a parameter
}
else if (resultState == CompareResultState.FolderContentEqualOwnStateDifferent)
{
 // Folder content is the same, folder's own state is different:
 // May occur if a folder-style module, e.g., FM 351, has equal subordinate elements but
 // the module itself is different, e.g., in a parameter
}
else if (resultState == CompareResultState.FolderContentsIdentical)
{
 ...
}
...
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.1.16 Setting password policy for PLC

Requirement

- The TIA Portal is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to set the security password policy for legacy and plus PLC.

Setting password policy for legacy PLC

You can use the LegacyPlcPasswordPolicyService at the project base object to set the security password policy for legacy PLC.

The LegacyPlcPasswordPolicyService feature supports the following attributes:

Attribute name	Data type	Description
PasswordPolicyEnabled	Boolean	If PasswordPolicyEnabled is "True" then all the other configured attribute values will be considered for validating the password. If PasswordPolicyEnabled is "False" then new policy cannot be applied for any legacy plc.
MinimumLength	Short	Sets the value of minimumLength of the password The allowed range value for minimum length is 5 to 8
MinimumNumericCharacter-Length	Short	Sets the value of minimumNumericCharacterLength in password The allowed range value for minimum numeric character length is 0 to 8
MinimumSpecialCharacter-Length	Short	Sets the value of minimumSpecialCharacterLength in password The allowed range value for minimum numeric character length is 0 to 8
IncludesLowerCaseAndUpper-CaseCharacters	Boolean	Sets the value of includesLowerCaseAndUpperCaseCharacters in password

A custom exception PasswordPolicySettingsException is thrown when you attempt to set the attribute value outside the permissible range.

Setting password policy for plus PLC

You can use the PlcPasswordPolicyService at the project base object to set the security password policy for plus PLC.

The PlcPasswordPolicyService feature supports the following attributes:

Attribute name	Data type	Description
PasswordPolicyEnabled	Boolean	If PasswordPolicyEnabled is "True" then the UMAC (i.e. Password complexity for project users) configured attribute values will be considered for validating the password If PasswordPolicyEnabled is "False" then new policy cannot be applied for any plus plc.

Exception

When the password is set from the Openness script for a protection access level, a new type of exception EngineeringPasswordPolicyViolationException will be thrown if the entered password does not comply with the policy.

Program code

To set the password policies for legacy PLC modify the following program code:

```
public void LegacyPlc_SecuritySettingPasswordPolicy_SetAttributes()
{
LegacyPlcPasswordPolicyService m_LegacyPlcPasswordPolicyService =
m_project.GetService<LegacyPlcPasswordPolicyService>();
m_LegacyPlcPasswordPolicyService.PasswordPolicyEnabled= true;
m_LegacyPlcPasswordPolicyService.MinimumLength = 6;
m_LegacyPlcPasswordPolicyService.MinimumNumericCharacterLength = 2;
m_LegacyPlcPasswordPolicyService.MinimumSpecialCharacterLength=2;
m_LegacyPlcPasswordPolicyService.IncludeLowerCaseAndUpperCaseCharacters = true;
}
```

To set the password policies for plus PLC modify the following program code:

```
public void PlusPlc_SecuritySettingPasswordPolicy_SetAttributes()
{
PlcPasswordPolicyService m_PlcPasswordPolicyService =
m_project.GetService<PlcPasswordPolicyService>();
m_PlusPlcPasswordPolicyService.PasswordPolicyEnabled= true;
}
```

5.11.1.17 Managing dynamic certificate settings

Requirement

- The TIA Portal is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to configure dynamic certificate management for all supported services at a central place within the TIA Portal instead of modifying each application and service individually.

You can use the Certificate Management Configuration service to manage dynamic certificate, which will be available for the PLC instance of all PLCs device family with version V3.0 and later.

Property

The CertificateManagementConfiguration supports the following properties:

Property name	Data type	Description	Access	Note
Usage	CertificateConfigurationUsage	Gets or sets the usage of the certificate management configuration The Usage property allows you to choose two options either certificates configured and download using TIA Portal or certificates provided by certificate management during runtime . The Usage property has two enum values: • TIAPortal • Runtime	Read/Write	The EngineeringTargetInvocationException is thrown in a set operation for the value CertificateConfigurationUsage.Runtime when WebServer and OpcUaServer are in deactivate state
CertificateExpirationEventActivated	Boolean	Enable or disable the certificate expiration setting of the certificate management configuration	Read/Write	The EngineeringTargetInvocationException is thrown when Usage property is set as certificates configured and download using TIA Portal
RemainingCertificateLifetime	ushort	Gets or sets the value for showing event for remaining certificate life time of the certificate management configuration This is a percentage value that allows to calculate the time after which a PLC alarm shall be raised to inform the user about an upcoming expiration.	Read/Write	The EngineeringTargetInvocationException is thrown when the EnableCertificateExpirationEvent property value is set as False or when the value is not set between 10 - 90
CertificateSupportedServices	CertificateSupportedServiceComposition	Gets all the certificate supported services of the certificate management configuration which enabled dynamic certificate management . The services currently supported are OPC UA and Webserver. The services currently supported are OPC UA and Webserver.	Read	

Method

The CertificateSupportedService() represents the supported services of the Certificate Management Configuration. It accepts the following property:

Property name	Data type	Description	Access	Note
ServiceType	CertificateSupportedServiceName	Represents name of the service supported by certificate management configuration. The CertificateSupportedServiceName accepts the following Enum item and its Enum value: • Webserver - 100 • OpcUaServer - 200	Read	
Id	ushort	Represents id of the service supported by certificate management configuration	Read	
ServiceGroupName	string	Represents group name of supported services of the certificate management configuration	Read/ Write	The EngineeringTargetInvocationException is thrown when the value of service is set to greater than 64 characters.

Program code

```
//...
// for getting the service
CertificateManagementConfiguration certifcateConfigurationService =
plc.GetService<CertificateManagementConfiguration>();
// To setup the usage
certifcateConfigurationService.Usage = CertificateConfigurationUsage.TIAPortal;
// for service count
var servicesCount = certifcateConfigurationService.CertificateSupportedServices.Count;
// For setting group name
certifcateConfigurationService.CertificateSupportedServices[0].ServiceGroupName =
"ServiceName_1";
```

Handle dynamic certificates during download to PLC

The SelectiveDeleteDownload can be used to handle certificates during download to PLC. Based on this option, dynamic certificates will be deleted.

The SelectiveDeleteDownload.CurrentSelection supports the following Enum item and its associated enum value:

- SelectiveDeleteDataSelections.AcceptAll - accept all
- SelectiveDeleteDataSelections.DeleteSelected - delete selected
- SelectiveDeleteDataSelections.DeleteAll - delete all

Program code: Handle dynamic certificates during download to PLC

```
private void DownloadConfigurationDelegate(DownloadConfiguration downloadConfiguration)
{
    if (downloadConfiguration is SelectiveDeleteDownload selectiveDeleteDownload)
    {
     selectiveDeleteDownload.CurrentSelection = SelectiveDeleteDataSelections.AcceptAll;
    }
}
```

5.11.2 Functions for accessing PLC service

5.11.2.1 Access level setting

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness API to access the Access level setting via the Hardware PlcAccessLevelProvider at the Deviceitem CPU of Device.

This feature provides actions to set / reset the password and a modeled attribute to set / read the access level setting.

- void SetPassword (PlcProtectionAccessLevel accessLevelType, SecureString password)
- void ResetPassword (PlcProtectionAccessLevel accessLevelType)
- attribute PlcProtectionAccessLevel

In general the handling via Openness is similar to the TIA Portal UI.

- PlcProtectionAccessLevel can always be read and set at S7-1200/1500 PLCs.
- If access level is set to anything different than Full access (or Full access incl. fail-safe) a full access password has to be set. Otherwise there will be a compile error: "Password must not be empty"
- It is not allowed to set the same password for different access levels
- It is only possible to set / reset passwords for higher levels than the one set as access level (if PlcProtectionAccessLevel is set to HMIAccess it is only possible to set a password for ReadAccess and FullAccess)

Possible exceptions:

- EngineeringTargetInvocationException: **"Same password already exists for other access level"** - will be thrown if the user tries to set the same password for different access levels
- EngineeringTargetInvocationException: **"Cannot set password for the access level"** - will be thrown if the user tries to set a password for a stricter access level than the one selected
- EngineeringTargetInvocationException: **"Password cannot be empty"** - will be thrown if the user tries to set an empty password
- EngineeringTargetInvocationException: **"Access Level is not valid"** - will be thrown if the user tries to set the access level to FullAccessIncludingFailsafe at non failsafe PLCs

Access level

In Openness the access level is modelled as an Enum named PlcProtectionAccessLevel. The name of the Enum fields are based on the "Access Level" entries for S7-1200/1500 PLCs.

TIA UI name	Enum entry	Value	Remarks
-	None	0	For enum initialization. This value must not be set by the Openness user.
Full access (no protection)	FullAccess	1	
Read access	ReadAccess	2	
HMI access	HMIAccess	3	
No access	NoAccess	4	
Full access incl. fail-safe (no protection)	FullAccessIncludingFailsafe	5	Only available at failsafe PLCs

An EOM Attribute(Modeled Attribute) named PlcProtectionAccessLevel of type PlcProtectionAccessLevel enum is available at PLCs to Set/Get the access level.

You can use the following code example for PlcProtectionAccessLevel with No access:

```
DeviceItem S71500PLC = ...;
PlcAccessLevelProvider myPlcAccessLevelProvider =
S71500PLC.GetService<PlcAccessLevelProvider>();
myPlcAccessLevelProvider.PlcProtectionAccessLevel = PlcProtectionAccessLevel.NoAccess;
```

Password

The password is implemented as SecureString for security reasons. It is only possible to set / reset the passwords for the different access levels using the corresponding action. Reading the passwords is not supported.

```
myPlcAccessLevelProvider.SetPassword(PlcProtectionAccessLevel.ReadAccess,
someSecureString);
myPlcAccessLevelProvider.ResetPassword(PlcProtectionAccessLevel.ReadAccess);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.2 Accessing Software Checksum

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application
  See Opening a project (Page 128)

Application

You can use the Openness API to access the software checksum for a PLC station.

Program code

To access, you get the instance of PlcSoftware and and load a service instance of PlcChecksumProvider.

```
PlcSoftware plc = …;
 //get PlcSoftware instance
PlcChecksumProvider checksumProvider =
plc.GetService<PlcChecksumProvider>();
```

In case the PLC does not support the checksum calculation then the GetService call returns null.

At the PlcChecksumProvider instance the software checksum can be reached as follow:

```
string softwareChecksum = checksumProvider.Software;
```

The Software attribute is read only and it returns null when the program is not compiled.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.3 Assigning PC interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

For the interface assignment, you have to get a PcInterfaceAssignment service from Siemens.HW.Features on the device item interface.

If the PC-Station version is <= 2.0, the methods of this service will return null and interface assignment will not be possible for the interfaces that are assigned to PC-Station or Windows only or none. For the interfaces that are assigned to SW-CPU, this is again possible.

Note

The support of PC Station is very limited in the CAx interchange. Only PC Stations with version 1.0 and the default value for InterfaceAssignment are supported. The reason is these two values have impact on device structure (what is built-in, what can be plugged where). But since the parameters are not part of the AutomationML file, the values are not set during import and as a result the structure of the IPC does not fit and the other modules can not be plugged.

Program code: Interface assignment

Modify and use the following program code for interface assignment:

```
DeviceItem swCpu = ...;
DeviceItem myInterface = ...;
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
//user has to give device item sw cpu in case he/she wants to assign to it. (in the future,
there could be multi sw-cpu cases)
provider.AssignInterface(PcInterfaceAssignmentMode.SoftwarePlc, swCpu);
...
provider.AssignInterface(PcInterfaceAssignmentMode.PcStation);
...
provider.AssignInterface(PcInterfaceAssignmentMode.None);
...
```

Modify and use the following program code to get information about the current assignment mode of an interface:

```
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
PcInterfaceAssignmentMode mode = provider.PcInterfaceAssignmentMode;
switch(mode)
{
case PcInterfaceAssignmentMode.None:  Console.WriteLine("Assigned to none or windows-
only.");
break;
case PcInterfaceAssignmentMode.PcStation: Console.WriteLine("Assigned to pc-station.");
break;
case PcInterfaceAssignmentMode.SoftwarePlc: Console.WriteLine("Assigned to SoftwarePlc: "
+ provider.SoftwarePlc.Name);
break;
}
```

Program code: Hardware Resource and IPC Expansion configuration

You can also select correct IPC Expansion and HardwareResource property via PcInterfaceAssignment sevice using the following code:

```
DeviceItem myInterface = ...;
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
//this method returns a subset of IpcExpansion values which are available for the IPC that
the interface is plugged in. IEnumerable<string> ipcExpansionChoices =
provider.GetAvailableIPCExpansions();
string myChoice;
foreach(var choice in ipcExpansionChoices)
{
//user must select the desired value depending on his/her configuration
if (choice.Contains("3yxx1")
{
myChoice = choice;
break;
}
}
provider.IpcExpansion = myChoice;
//After assigning IpcExpansion value, user may assign HardwareResource value using the
following enumeration.
provider.HardwareResource = HardwareResource.X101;
//OR
//myInterface.SetAttribute("HardwareResource", HardwareResource.X101);
//myInterface.SetAttribute("IpcExpansion", mychoice);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.4 Determining the status of a PLC

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can determine the state of a PLC or all PLCs in a project.

TIA Portal Openness distinguishes between the following states:

- Offline
- PLC is connected ("Connecting")
- Online
- PLC is disconnected ("Disconnecting")
- Incompatible
- Not reachable
- Protected

Program code

Modify the following program code to determine the state of a PLC:

```
public static OnlineState GetOnlineState(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    return onlineProvider.State;
}
```

Modify the following program code to determine the state of all PLCs in a project:

```
public static void DetermineOnlineStateOfAllProjectDevices(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                OnlineState state = onlineProvider.State;
            }
        }
    }
}
```

5.11.2.5 Accessing parameters of an online connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness API interface to determine or set parameters for an online connection:

- Enumerate the available connection modes to a PLC
- Enumerate the available interfaces to a PLC
- Enumerate the allocated slots
- Enumerate the available addresses of the subnets and gateways
- Set the connection parameters.

For information on IsSecureCommunication, refer Supporting secure S7 communication TLS (Page 475)

Program code: Determining connection parameters

Modify the following program code to enumerate the available connection modes, PC interfaces and slots:

```
public static void EnumerateConnectionModesOfPLC(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null)
    {
       return; // Only cpu device items can provide OnlineProvider service
    }
    // Accessing connection configuration object
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    // Now access connection configuration members
    foreach (ConfigurationMode mode in configuration.Modes)
    {
        Console.WriteLine("Mode name:{0}", mode.Name);
        foreach (ConfigurationPcInterface pcInterface in mode.PcInterfaces)
        {
            Console.WriteLine("PcInterface name:{0}", pcInterface.Name);
            Console.WriteLine("PcInterface number:{0}", pcInterface.Number);
            foreach (ConfigurationTargetInterface targetInterface in
pcInterface.TargetInterfaces)
            {
                Console.WriteLine("TargetInterface:{0}", targetInterface.Name);
            }
        }
    }
}
```

You can also access a connection mode and a PC interface by name:

```
public static ConfigurationTargetInterface
GetTargetInterfaceForOnlineConnection(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find("PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    return slot;
}
```

Modify the following program code to enumerate the addresses of the subnets and gateways available on a PC interface:

```
public static void EnumeratingPCInterfaceSubnetsAndGateways(ConfigurationPcInterface
pcInterface)
{
    foreach (ConfigurationSubnet subnet in pcInterface.Subnets)
    {
        Console.WriteLine("Subnet name:{0}", subnet.Name);
        foreach (ConfigurationGateway gateway in subnet.Gateways)
        {
            //Get the name of the gateway:
            Console.WriteLine("Gateway name:{0}", gateway.Name);
            //Get the IP address of each gateway:
            foreach (ConfigurationAddress gatewayAddress in gateway.Addresses)
            {
                Console.WriteLine("Gateway Address:{0} has {1}", gatewayAddress.Name,
gatewayAddress.Address);
            }
        }
    }
}
```

You can also access subnets and gateways by their name or IP address:

```
public static void AccessSubnetAndGatewayOfPCInterface(ConfigurationPcInterface
pcInterface)
{
    ConfigurationSubnet subnet = pcInterface.Subnets.Find("PN/IE_1");
    ConfigurationAddress subnetAddress = subnet.Addresses.Find("192.168.0.1");
    ConfigurationGateway gateway = subnet.Gateways.Find("Gateway 1");
    ConfigurationAddress gatewayAddress = gateway.Addresses.Find("192.168.0.2");
}
```

Program code: Setting connection parameters

Note

All the connection parameters previously set are overwritten when you set the connection parameters. If you have already set the connection parameters directly in the TIA Portal, it is not necessary to call `ApplyConfiguration`. If there is already an online connection to a PLC while `ApplyConfiguration` is called, an exception is thrown.

Modify the following program code to set slot parameters:

```
public static void SetConnectionWithSlot(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    configuration.ApplyConfiguration(slot);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

Modify the following program code to set gateway address parameters:

```
public static void SetConnectionWithGatewayAddress(OnlineProvider onlineProvider, string
subnetName, string gatewayAddressName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddress gatewayAddress = subnet.Addresses.Find(gatewayAddressName);
    configuration.ApplyConfiguration(gatewayAddress);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

Modify the following program code to set subnet address parameters:

```
public static void SetConnectionWithSubnetAddress(OnlineProvider onlineProvider, string
subnetName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddressComposition addresses = subnet.Addresses;
    configuration.ApplyConfiguration(addresses[0]);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

See also

Supporting secure S7 communication TLS (Page 475)

5.11.2.6 Accessing fingerprint for quick station compare

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

You can use the TIA Portal Openness to get FingerprintData for different aspects of PLC device configuration which can be used to realize a quick station compare. It is done through FingerprintDataProvider service that can be acquired from a given TIA Portal. An instance of FingerprintDataProvider will be returned on GetService call, else the service will return null.

Program code: Retrieving FingerprintDataProvider from TIA Portal

```
TiaPortal tia = new TiaPortal(TiaPortalMode.WithUserInterface);
FingerprintDataProvider fingerprintDataProvider =
tia.GetService<FingerprintDataProvider>();
if (fingerprintDataProvider != null)
{
 ...
}
```

Parameters of FingerprintDataProvider method

To get the fingerprintData of a device, you have to call GetFingerprintData method of FingerprintDataProvider.

The following attributes are supported in FingerprintDataProvider:

Parameter name	Type	Description
configurationAddress	Siemens.Engineering.Online.Configuration	Address of device for which the fingerprintData have to be fetched.
onlineConfigurationDelegate	Siemens.Engineering.Online	Delegate that will be called to check configuration before fetching the fingerprintData.

Configuration Address

You should provide a ConfigurationAddress object to the GetFingerprintData. The address object will be used to establish a connection to the device for which the fingerprintData have to be fetched. The ConfigurationAddress object must be created in the ConnectionConfiguration of the FingerprintDataProvider. For information on IsSecureCommunication, refer Supporting secure S7 communication TLS (Page 475)

For example, you can use the following code to create an address object:

```
...
ConnectionConfiguration configuration = fingerprintDataProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
// Create an address. This "ConfigurationAddress" is used as parameter for getting the
fingerprintData
ConfigurationAddress fingerprintAddress = pcInterface.Addresses.Create("192.68.0.1");
...
```

The Configuration contains a list of supported Modes. You have to select one of the Modes that should be used to get fingerprints. The selected ConfigurationMode contains a list of all local PCInterfaces that support the selected Mode. You have select one of the interfaces. The desired address can be created in the Address collection of the selected ConfigurationPcInterface.

Program code: Retrieving the FingerprintData

You can start retrieving the FingerprintData of the station by calling the action GetFingerprintData. The following parameters are supported:

Parameter	Description
ConfigurationAddress	The address of the device
OnlineConfigurationDelegate	The callback to handle inhibits)

```
...
OnlineConfigurationDelegate preConfigurationDelegate = PreConfigureFingerprint;
FingerprintDataResult result =
fingerprintDataProvider.GetFingerprintData(fingerprintAddress , preConfigurationDelegate);
// The fingerprints
FingerprintDataItemComposition allFingerprints = result.FingerprintDataItems;
foreach (FingerprintDataItem item in allFingerprints)
{
// do something with the fingerprint ...
}
internal void PreConfigureFingerprint(OnlineConfiguration onlineConfiguration)
{
...
}
```

Online configuration delegate

The possible online configuration types are listed below:

Configuration name	Description and properties
OnlineConfiguration	• Base class for other configurations • OnlineConfiguration.Message: string (read only property that contains the configuration message)

Because this is the Base class for all other configurations, this property adapter is therefor available in all configurations.

The datatype of the configuration is given below:

Configuration	DataType	Description and Action
OnlineConfiguration	OnlineReadAccessPassword	Set password via SetPassword(password:SecureString) method. Enter a password to gain read access to the module

Unhandled configuration that can prevent accessing fingerprint causes EngineeringDelegateInvocationException and aborts action. An EngineeringDelegateInvocationException will be thrown in case of an unhandled exception within the Delegate.

PreConfigureFingerprint implementation example

```
private static void PreConfigureFingerprint(OnlineConfiguration onlineConfiguration)
{
OnlineReadAccessPassword readAccess = onlineConfiguration as OnlineReadAccessPassword;
if (readAccess != null)
{
string passWD = "passWD";
var password = new SecureString();
foreach (var c in passWD) password.AppendChar(c);
readAccess.SetPassword(password);
return;
}
throw new NotSupportedException(); // Exception thrown in the delagate will cancel get
fingerprints
}
```

FingerprintDataResult

The FingerprintDataResult returned by the GetFingerprintData action provides a composition of all the fingerprints defined for the device.

For example, you can use the below code to examine fingerprint items:

```
{
...
FingerprintDataResult result = dataProvider.GetFingerprintData(fingerprintAddress,
preConfigurationDelegate);
FingerprintDataItemComposition allFingerprints = result.FingerprintDataItems;
foreach (FingerprintDataItem item in allFingerprints)
{
string fingerprintDataIdentifier = item.FingerprintDataIdentifier;
string fingerprintDataValue = plcItem.FingerprintDataValue;
// Do something, e.g. store Identifier and Value in a file
}
}
```

Error Handling within the Openness fingerprints

All errors within the Openness fingerprints will be mapped as recoverable user exceptions EngineeringTargetInvocationException. So any error in an Openness action will not lead to a TIA-Portal crash.

Precondition is, that the client handle the exception. For example this could happened like shown in following code

```
TiaPortal currentTIA = ...;
FingerprintDataProvider dataProviderFingerprints =
currentTIA.GetService<FingerprintDataProvider>();
OnlineConfigurationDelegate configurationDelegate = OnlineConfigurationFingerprintData;
try
{
// The used addr referes a PLC300.
FingerprintDataResult dataResult = dataProviderFingerprints.GetFingerprintData(addrObj,
configurationDelegate);
// Try to get a list of all FingerprintDataItems. But a PLC300 supports no fingerprints.
FingerprintDataItemComposition dataItems = dataResult.FingerprintDataItems;
foreach (FingerprintDataItem searchItem in dataItems)
{
string valueToCompareIdent = searchItem.FingerprintDataIdentifier;
string valueToCompareValue = searchItem.FingerprintDataValue;
DoAnything(valueToCompareIdent, valueToCompareValue);
}
}
catch (EngineeringTargetInvocationException targetException)
{
// In case of the not supported PLC this exception will be catched.
Console.WriteLine(targetException.Message);
}
```

Helper classes

For this example two classes are needed. You can use the below program code to encapsulate fingerprint data pairs.

```
/// <summary>
/// Helperclass to handle a "Dictionary" as serializeable class object
/// </summary>
[Serializable]
public class FingerprintDataPairs : Dictionary<string, string>, ISerializable
{
public FingerprintDataPairs(){}
public FingerprintDataPairs(SerializationInfo si, StreamingContext context)
{
KeyValuePair<string, string>[] dataPair = si.GetValue("KeyValuePairs",
typeof(KeyValuePair<string, string>[])) as KeyValuePair<string, string>[];
if (dataPair != null)
foreach (KeyValuePair<string, string> pair in dataPair)
this.Add(pair.Key, pair.Value);
}
}
```

You can use the below example to handle the fingerprint data and compare:

```
/// <summary>
/// Helperclass for handle the fingerprint data and compare two datasets.
/// </summary> public class FingerprintDataCompare
{
internal FingerprintDataPairs m_FingerprintDataPairs;
internal FingerprintDataPairs FingerprintDataPairSet
{
get
{
if (m_FingerprintDataPairs == null) m_FingerprintDataPairs = new FingerprintDataPairs();
return m_FingerprintDataPairs;
}
set
{
m_FingerprintDataPairs = value;
}
}
internal bool CompareDataSets(FingerprintDataPairs dataSet1, FingerprintDataPairs dataSet2)
{
foreach (string key in dataSet1.Keys)
{
if (dataSet2[key] != dataSet1[key])
return false;
}
return true;
}
internal void SaveFingerprintDataItems(FingerprintDataPairs fingerprintDataPairSet, string
filename)
{
using (FileStream fs = new FileStream(filename, FileMode.Create))
{
BinaryFormatter formatter = new BinaryFormatter();
try
{
formatter.Serialize(fs, fingerprintDataPairSet);
}
catch (SerializationException e)
{
Console.WriteLine("Failed to write fingerprints. See: " + e.Message);
}
finally
{
fs.Close();
}
}
}
internal FingerprintDataPairs LoadFingerprintDataItems(string filename)
{
FingerprintDataPairs dataPair = null;
using(FileStream fs = new FileStream(filename, FileMode.Open))
{
BinaryFormatter formatter = new BinaryFormatter();
try
{
dataPair = (FingerprintDataPairs)formatter.Deserialize(fs);
```

```
}
catch (System.Runtime.Serialization.SerializationException e)
{
Console.WriteLine("Failed to read fingerprints. See: " + e.Message);
}
finally
{
fs.Close();
}
}
return dataPair;
}
}
```

Compare Fingerprints

To compare, you can use the FingerprintDataResult

```
...
// See chapter "FingerprintDataResult"
FingerprintDataResult fingerprintDataResult = dataProvider.GetFingerprintData(address,
preConfigurationDelegate);
...
```

To save the Fingerprints. you can use the Helper Classes:

```
string pathDat = "FingerprintDataOverview.dat";
// Create a fingerprint comparer and fill the dataitems with the result of
"GetFingerprintData".
FingerprintDataCompare fingerprintDataSet = new FingerprintDataCompare();
foreach (FingerprintDataItem item in fingerprintDataResult.FingerprintDataItems)
fingerprintDataSet.FingerprintDataPairSet.Add(item.FingerprintDataIdentifier,
item.FingerprintDataValue);
fingerprintDataSet.SaveFingerprintDataItems(fingerprintDataSet.FingerprintDataPairSet,
pathDat);
```

With saved FingerprintDataItems and a current FingerprintDataResult a Compare of Fingerprints can be realized

```
...
string pathDat = "FingerprintDataOverview.dat";
FingerprintDataCompare fingerprintDataSet = new FingerprintDataCompare();
foreach (FingerprintDataItem item in fingerprintDataResult.FingerprintDataItems)
fingerprintDataSet.FingerprintDataPairSet.Add(item.FingerprintDataIdentifier,
item.FingerprintDataValue);
FingerprintDataPairs loadDataSet = new FingerprintDataPairs();
loadDataSet = fingerprintDataSet.LoadFingerprintDataItems(pathDat);
if (!fingerprintDataSet.CompareDataSets(loadDataSet,
fingerprintDataSet.FingerprintDataPairSet))
{
// Here is place to implement user actions.
}
```

5.11.2.7 Accessing Cross Reference Service on Step7

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
-

Application

You can use the TIA Portal Openness to access the CrossReferenceService through GetService method on Step7 objects. The Cross Reference provides IEngineeringService i.e. CrossReferenceService on applicable Step7 objects. A null value is being returned from the TIA Portal Openness for non-applicable Step7 objects trying to access CrossReferenceService through GetService method.

The following Step7 objects that provide Cross Reference Openness support:

- OB
- FB
- FC
- DB
- InstanceDB
- GlobalDB
- Array DB
- PlcTag

- PlcSystemConstant
- Plc User Data Type

Program code

```
Software softTarget = container.Software; //Accessing lower levels until we reach the
PlcBlockComposition
PlcSoftware plcSoft = (PlcSoftware)softTarget;
PlcBlockSystemGroup blockComposition = plcSoft.BlockGroup;
PlcBlockComposition blockFolder = blockComposition.Blocks;
PlcBlock mainBlock = blockComposition.Blocks.Find("Main");
if (mainBlock != null)
{
try
{
CrossReferenceService crossReferenceService =
mainBlock.GetService<CrossReferenceService>();
......
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

5.11.2.8 Getting Cross References for Step7

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to provide Cross Reference information on applicable Step7 objects. The Openness API returns CrossReferenceResult which provide the textual values of cross reference user interface and actual IEngineeringObject of applicable Step7 objects.

The cross reference will attempt to provide the Step7 object itself on which you can perform additional actions supported by the respective TIA Portal Openness objects.

Note

Currently, the members of Step7 object does not support the type IEngineeringObject i.e. (Step7 object members - DB member, FB Members, FC Members , Tag Members, UDT members) but CrossReferenceResult includes all the textual information that is already available.

Properties

Below table provides the list of properties supported by Step7 and WinCC Unified objects in TIA Portal Openness:

Custom type	Property name	Datatype	Access Read/Write
SourceObject	Name	String	Read
	Address	String	Read
	TypeName	String	Read
	Device	String	Read
	Path	String	Read
	UnderlyingObject	Object	Read
ReferenceObject	Name	String	Read
	Address	String	Read
	TypeName	String	Read
	Device	String	Read
	Path	String	Read
	UnderlyingObject	Object	Read
Location	Name	String	Read
	ReferenceLocation	String	Read
	ReferenceType	Enum	Read
	Access	Enum	Read
	ReferencedAs	Object	Read
	ReferncedAsName	String	Read
	Address	String	Read
	TypeName	String	Read

Below table describes the list of navigators supported by Step7 and WinCC Unified objects in TIA Portal Openness:

Custom type	Navigators	Datatype
CrossReferenceResult	Sources	SourceObjectComposition
SourceObject	References	ReferenceObjectComposition
ReferenceObject	Locations	LocationComposition

Below table describes the list of methods and enums supported by Step7 and WinCC Unified objects in TIA Portal Openness:

Function Name with Signature	Enum
public CrossReferenceResult GetCrossReferences(CrossReferenceFilter filter)	CrossReferenceFilter
	ReferenceType
	Access

Below table describes the list of filter enums supported by Step7 and WinCC Unified objects in TIA Portal Openness:

Enum	Values
AllObjects	Show All Object
ObjectsWithReferences	Show Object With References
ObjectsWithoutReferences	Show Object Without References
UnusedObjects	Show Unused Objects

Program code

```
Public void GetCrossReferences()
{
CrossReferenceService crossReferenceService  = MainBlock.GetService<CrossReferenceService
>();   CrossReferenceResult rootResultObject =
xRefService.GetCrossReferences(CrossReferenceFilter.AllObjects);
SourceObjectComposition rootResultSourceObjects = rootResultObject.Sources; //The API
returns a single source object but the type is a composition keeping LTS in mind.
PrintSourceObjects(rootResultSourceObjects);
}
Private void PrintSourceObjects(SourceObjectComposition sourceObjects)
{
foreach(SourceObject source in sourceObjects)
{
// In this approach, cross reference tries to provide the actual client openness object via
the "UnderlyingObject"attribute.
// However, due to the fact that cross reference result can have an object of any type such
as OB, FB, DB, Tag etc returned from the CrossReferenceService
// and a limitation in openness that the type cannot be pre-mapped, the user must cast the
object before accessing its attributes.
// As a result, the openness user must convert sourceObject.UnderlyingObject to required
EOM type (such as OB, DB, FB, Tag etc)
if (source.UnderlyingObject is DataBlock)
{
var db = source.UnderlyingObject as DataBlock;//Perform actions on db.
}
else if(source.UnderlyingObject is CodeBlock)
{
var cb = source.UnderlyingObject as CodeBlock;
//Perform actions on cb.
}
// Every new type for which support is added, the openness user must add the casting logic
as shown in the above if-else conditional statement.
// Currently, the EOM support is NOT available for Member objects such as Datablock member,
Tag member etc.
// In such cases, the "source.UnderlyingObject" will be null.
Console.WriteLine(source.Name);
//No other attributes are available from the IEngineeringObject. All the remaining values
must be given from cross reference separately via SourceObject EOM.
Console.WriteLine(source.Address);
Console.WriteLine(source.TypeName);
Console.WriteLine(source.Device);
Console.WriteLine(source.Path);
//Get the source object references
ReferenceObjectComposition referenceObjects = source.References;
PrintReferences(referenceObjects);
//Get the source object's children
SourceObjectComposition sourceObjectChildren = source.Children;
PrintSourceObjects(sourceObjectChildren);
}
}
private void PrintReferences(ReferenceObjectComposition referenceObjects)
{
foreach (ReferenceObject referenceObject in referenceObjects)
{
// Needs similar casting for each type as seen in source object example snippet
```

```
if (referenceObject.UnderlyingObject is DataBlock)
{
var db = referenceObject.UnderlyingObject as DataBlock;
//Perform actions on db.
}
else if(referenceObject.UnderlyingObject is CodeBlock)
{
var cb = referenceObject.UnderlyingObject as CodeBlock;
//Perform actions on cb.
}
//No other attributes are available from the IEngineeringObject. All the remaining values
must be given from cross reference separately via ReferenceObject EOM.
Console.WriteLine(referenceObject.Name);
Console.WriteLine(referenceObject.Address);
Console.WriteLine(referenceObject.TypeName);
Console.WriteLine(referenceObject.Device);
Console.WriteLine(referenceObject.Path);
LocationComposition locations = referenceObject.Locations;
//Looping through ReferenceLocations
foreach (Location location in locations)
{
Console.WriteLine(location.Name);
Console.WriteLine(location.ReferenceLocation);
Console.WriteLine(location.ReferenceType);
Console.WriteLine(location.Access);
if(location.ReferencedAs is Tag)
var referencedAsObject = location.ReferencedAs as Tag;
}
// Needs similar casting for each type as seen in source object example snippet
Console.WriteLine(location.ReferncedAsName);
Console.WriteLine(location.Address);
Console.WriteLine(location.TypeName);
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.9 Setting PLC online of R/H system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

You can use the RHOnlineProvider service to set online either to primary PLC or backup PLCs of R/H system.

Program code: Accessing RHOnlineProvider service from a device

Modify the following code to access RHOnlineProvider:

```
Device device = project.Devices.Find("S7-1500R/H-System_1");
RHOnlineProvider rhOnlineProvider = device.GetService<RHOnlineProvider>();
```

Program code: Setting connection parameters

You can use ConnectionConfiguration object to set a connection to the device. It can be accessed from Configuration property of the RHOnlineProvider. For more information about how to set connection, Refer Accessing parameters of an online connection (Page 434)

Modify the following program code to set a connection mode and access a PC interface by name. For information on IsSecureCommunication, refer Supporting secure S7 communication TLS (Page 475)

```
ConnectionConfiguration connectionConfiguration = rhOnlineProvider.Configuration;
ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit
Ethernet", 1);
ConfigurationTargetInterface targetConfiguration = pcInterface.TargetInterfaces.Find("1
X1");
bool success = connectionConfiguration.ApplyConfiguration(targetConfiguration);
```

Note

R/H system consists of two PLCs, a single connection configuration is provided to you.

Program code: Setting online R/H system

You can set online either to primary or backup PLC. The user attempts to set online to both targets simultaneously will encounter EngineeringTargetInvocationException from system.

Modify the following program code to set online to the primary PLC:

```
OnlineState onlineState = rhOnlineProvider.GoOnlineToPrimary();
```

Modify the following program code to set online to the backup PLC:

```
OnlineState onlineState = rhOnlineProvider.GoOnlineToBackup();
```

Note

You are allowed to reuse previously stored password when setting a PLC Online of R/H system.

Program code: Determining online status of R/H system

You can use PrimaryState and BackupState properties of RHOnlineProvider to determine the online connection status of primary PLC and backup PLC individually . Both properties return enum OnlineState. For more information on identify the online state of PLC, Refer Determining the status of a PLC (Page 433)

Modify the following program code to determine the state of primary PLC and backup PLC:

```
RHOnlineProvider rhOnlineProvider = ...;
OnlineState primaryState = rhOnlineProvider.PrimaryState;
OnlineState backupState = rhOnlineProvider.BackupState;
```

Program code: Setting offline R/H system

Modify the following program code to set a currently online R/H system to an offline state by invoking RHOnlineProvider.GoOffline method:

```
rhOnlineProvider.GoOffline();
```

See also

Accessing parameters of an online connection (Page 434)

Determining the status of a PLC (Page 433)

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Supporting secure S7 communication TLS (Page 475)

5.11.2.10 Accessing software container from primary PLC of R/H system

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

You can use primary PLC device of an R/H system to access software container, for example the R/H system will provide software container for a primary PLC device item representing PLC_1. Otherwise, it will provide null if you try to access a software container for backup PLC device representing PLC_2.

The specific of SoftwarContainer and its software property are described in Access software target. (Page 147)

Program code: Accessing software container

Modify the following program code to access software container from primary device of an R/H system:

```
foreach (DeviceItem deviceItem in rhDevice.DeviceItems)
{
  if (deviceItem.Name == "PLC_1")
  {
    SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
    ... //Work with softwareContainer
  }
}
```

See also

Access software target (Page 147)

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.11 Downloading PLCs of R/H System

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness application to download the primary and backup PLCs of R/H system. You candownload both hardware and software components of the system. (Refer Downloading hardware and software components to PLC device (Page 385))

Program code: Retrieving RHDownloadProvider

You can download to R/H system through RHDownloadProvider service from a device.

Modify the following program code to retrieve RHDownloadProvider:

```
...
Device device = project.Devices.Find("S7-1500R/H-System_1");
RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
...
```

Note

The DownloadProvider service will not be accessed for CPUs that are part of R/H system.

Program code: Retrieving IConfiguration

RHDownloadProvider provides ConnectionConfiguration object through Configuration property which will be used to configuring the connection to the device. For information on IsSecureCommunication, refer Supporting secure S7 communication TLS (Page 475)

Modify the following program code to retrieve IConfiguration object from ConnectionConfiguration on RHDownloadProvider:

```
...
RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
ConnectionConfiguration connectionConfiguration = rhDownloadProvider.Configuration;
ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit
Ethernet", 1);
IConfiguration targetConfiguration = pcInterface.TargetInterfaces.Find("1 X1");
...
```

Note

R/H systems consist of two PLCs, only one connection configuration object is provided that can be used for both primary and backup downloads.

Program code: Downloading primary CPU and backup CPU

Modify the following program code to download to the primary CPU by invoking RHDownloadProvider.DownloadToPrimary:

```
DownloadResult DownloadToPrimary(configuration, preDownloadConfigurationDelegate,
postDownloadConfigurationDelegate, downloadOptions);
```

Modify the following program code to download to the backup CPU by invoking RHDownloadProvider.DownloadToBackup:

```
DownloadResult DownloadToBackup(configuration, preDownloadConfigurationDelegate,
postDownloadConfigurationDelegate, downloadOptions);
```

Parameters of RHDownloadProvider method

Both RHDownloadProvider.DownloadToPrimary and RHDownloadProvider.DownloadToBackup accept the same parameters and also return a DownloadResult. For more information about the details of IConfiguration, DownloadConfigurationDelegate, DownloadOptions and DownloadResult, Refer Downloading hardware and software components to PLC device (Page 385)

Parameter name	Type	Description
configuration	Siemens.Engineering.Connection.IConfiguration	Connection configuration to a device.
onlineAddress	Siemens.Engineering.Connection.ConfigurationAddress	Optional parameter for download to a changed IP address
preDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate that will be called to check configuration before download
postDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate that will be called to check configuration after download
downloadOptions	Siemens.Engineering.Download.DownloadOptions	Download options

Depending upon the state of R/H system, you might request to stop the system for the download via DownloadConfigurations. Therefore, in addition to the configuration described

in Downloading hardware and software components to PLC device (Page 385), the following data type are added to support RHDownload.

Configuration	Data Type	Action	Description
DownloadSelection-Configuration	StopHSystem	Set CurrentSelection:StopHSystemSelections. Available enum values: • NoAction (No Action) • StopHSystem (Stop R/H-System)	The modules are stopped for downloading to device
	StopHSystemOrModule	Set CurrentSelection:StopHSystemOrModuleSelections. Available enum values: • NoAction (No action) • StopHSystem (Stop R/H-System) • StopModule (Stop module)	The modules are stopped for downloading to device
	StartBackupModules	Set CurrentSelection:StartBackupModulesSelections. Available enum values: • NoAction (No action) • SwitchToPrimaryCpu (Change to Primary)) • StartModule (Start module)	Start modules after downloading to device
	SwitchBackupToPrimary	Set CurrentSelection:SwitchBackupToPrimarySelections. Available enum values: • NoAction (No action) • SwitchToPrimaryCpu (Change to Primary)	Start modules after downloading to device.

Program code: Handling download configuration callbacks

Modify the following program code to DownloadToPrimary and DownloadToBackup invocations while handling configurations in the callbacks:

Download invocation example

```
static void Main(string[] args)
{
...
Project project = tiaPortal.Projects[0];
Device device = project.Devices.Find("S7-1500R/H-System_1");
RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
ConnectionConfiguration connectionConfiguration = rhDownloadProvider.Configuration;
ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit
Ethernet", 1);
IConfiguration targetConfiguration = pcInterface.TargetInterfaces.Find("1 X1");
 // Download to primary
DownloadResult primaryDownloadResult =
rhDownloadProvider.DownloadToPrimary(targetConfiguration,
PreConfigureDownloadCallback,
PostConfigureDownloadCallback,
DownloadOptions.Hardware | DownloadOptions.Software);
WriteDownloadResults(primaryDownloadResult);
// Download to backup
DownloadResult backupDownloadResult =
rhDownloadProvider.DownloadToBackup(targetConfiguration,
PreConfigureDownloadCallback,
PostConfigureDownloadCallback,
DownloadOptions.Hardware | DownloadOptions.Software);
WriteDownloadResults(backupDownloadResult);
...
}
private static void PreConfigureDownloadCallback(DownloadConfiguration
downloadConfiguration)
{
StopHSystem stopHSystem = downloadConfiguration as StopHSystem;
if (stopHSystem != null)
{
stopHSystem.CurrentSelection = StopHSystemSelections.StopHSystem;
}
OverwriteTargetLanguages overwriteTargetLanguages = downloadConfiguration as
OverwriteTargetLanguages; if (overwriteTargetLanguages != null)
{
overwriteTargetLanguages.Checked = true;
}
AlarmTextLibrariesDownload alarmTextLibraries = downloadConfiguration as
AlarmTextLibrariesDownload;
if (alarmTextLibraries != null)
{
alarmTextLibraries.CurrentSelection =
AlarmTextLibrariesDownloadSelections.ConsistentDownload;
return;
}
CheckBeforeDownload checkBeforeDownload = downloadConfiguration as CheckBeforeDownload;
if (checkBeforeDownload != null)
{
checkBeforeDownload.Checked = true;
return;
}
```

Download invocation example

```
ConsistentBlocksDownload consistentBlocksDownload = downloadConfiguration as
ConsistentBlocksDownload;
if (consistentBlocksDownload != null)
{
consistentBlocksDownload.CurrentSelection =
ConsistentBlocksDownloadSelections.ConsistentDownload;
return;
}
OverwriteSystemData overwriteSystenData = downloadConfiguration as OverwriteSystemData;
if (overwriteSystenData != null)
{
overwriteSystenData.CurrentSelection = OverwriteSystemDataSelections.Overwrite;
return;
}
}
private static void PostConfigureDownloadCallback(DownloadConfiguration
downloadConfiguration)
{
StartModules startModules = downloadConfiguration as StartModules;
if (startModules != null)
{
startModules.CurrentSelection = StartModulesSelections.StartModule;
return;
}
}
private static void WriteDownloadResults(DownloadResult result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);
RecursivelyWriteMessages(result.Messages);
}
private static void RecursivelyWriteMessages(DownloadResultMessageComposition messages,
string indent = "")
{
indent += "\t";
foreach (DownloadResultMessage message in messages)
{
Console.WriteLine(indent + "DateTime: " + message.DateTime);
Console.WriteLine(indent + "State: " + message.State);
Console.WriteLine(indent + "Message: " + message.Message);
RecursivelyWriteMessages(message.Messages, indent);
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Downloading to PLC devices (Page 385)

Supporting secure S7 communication TLS (Page 475)

5.11.2.12 Establishing or disconnecting the online connection to the PLC

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- All devices are enumerated.
  See Accessing device items (Page 360).

Application

You can establish the online connection to a PLC, or disconnect an existing online connection.

Program code

Modify the following program code to establish or disconnect the online connection to a PLC:

```
public static void SetOnlineConnection(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null) { return; }
    // Go online
    if (onlineProvider.Configuration.IsConfigured)
    {
        onlineProvider.GoOnline();
    }
   // Go offline
   onlineProvider.GoOffline();
}
```

You can also establish or disconnect the online connections to all available PLCs in a project.

```
public static void SetOnlineConnectionForAllPLCs(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                // Establish online connection to PLC:
                onlineProvider.GoOnline();

                // ...

                // Disconnect online connection to PLC:
                onlineProvider.GoOffline();
            }
        }
    }
}
```

5.11.2.13 Assigning project language to PLC

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal Openness to assign project languages to web server and display languages of an S71500 PLC. To achieve that a StructuredData type dynamic attributes is required.

To access multilingual settings the new dynamic attribute MultilingualSupport of type TableData has been added. This can be split up into Rows. At each Row the assigned project language can be set or read with the dynamic attribute ProjectLanguage.

Program code

Modify the following program code to assign project language to PLC:

```
//To Set the Project Language
LanguageSettings languageSettings = project.LanguageSettings;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
DeviceItem Plc = …;
Tabledata multilingualSupportTable = Plc.GetAttribute("MultilingualSupport");
StructuredDataComposition structuredDataComp = multilingualSupportTable.Rows; // This will
return a collection of Structured Data.
StrcuturedData firstRow= structuredDataComp[0]; //First row
Language activeGermanLanguage = activeLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
firstRow.SetAttribute("ProjectLanguage", activeGermanLanguage.Culture);
 //To Get the Languages of device displayed
var assignedToGerman = firstRow.GetAttribute("DisplayLanguage");
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.14 Assigning watch & force tables for web server and PLC display

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal Openness to assign already created watch and force tables to the web server. The watch table and force table are software that can be found, created, and deleted in the software container. You can use the TIA Portal Openness to export/ import Watch and Force tables. For export/import Watch/Force tables, please see Export/Import Watch & Force Table (Page 1385)

To assign the watch and force tables to the web server, the WatchAndForceTableAccessManager service is used at the PLC DeviceItem.

The service contains two navigators for watch and force tables.

- WatchtableAccessRules {WatchtableAccessRuleComposition}
- ForcetableAccessRules {ForcetableAccessRuleComposition}

The navigator WatchTableAccessRules provides the WatchTableAccessRuleComposition which contains objects of type WatchTableAccessRule. The composition is empty by default. For the WatchTableAccessRule objects, the actions Find and Create are defined.

WatchTable Access Rule & ForceTable AccessRule

- **PlcWatchTable** watchtable {get;} returns the software object watchtable. In order to exchange an assigned watch table, the user has to remove the assignment of the current one (using WatchTableAccessRule.Delete()) and add a new one (using WatchTableAccessRule.Create()).
- **WatchAndForceTableAccess** access {get; set;} returns / sets the web server access level whether it is read or read/write

TIA UI name	value	Openness enum description
-	0	None
Read	1	Read
Read/Write	2	Write

Error will be thrown in the following cases:

- If the web server has not been activated (WebserverActivate == FALSE), a EngineeringTargetInvocationException is thrown with error details stating that a WatchTableAccessRule can be added only if the web server is enabled.
- If the user tries to add a watch table with WatchAndForceTableAccess.None a ConfigOpenessUserException with error details stating that WatchAndForceTableAccess.None is not allowed as a access level

Program code: Assigning & unassigning watch table

Modify the following program code to search watch table in the web server:

```
WatchAndForceTableAccessManager mngr =
deviceItem.GetService<WatchAndForceTableAccessManager>();
WatchTableAccessRuleComposition watchTableCmp = mngr.WatchTableAccesseRules;
WatchTableAccessRule accessRule1 = watchTableCmp.Find(watchTable1);
```

Note

Null will be returned if that watchtable is not associated with any WatchTableAccessRule in the Webserver

Modify the following program code to create a new watch table to the web server with read access:

```
WatchAndForceTableAccessManager mngr =
deviceItem.GetService<WatchAndForceTableAccessManager>();
WatchTableAccessRuleComposition watchTableCmp = mngr.WatchTableAccesseRules;
watchTableCmp.Create(watchTable1, WatchAndForceTableAccess.Read);
```

Modify the following program code to delete an existing WatchTableAccessRule and unassign the watch table from the web server:

```
WatchTableAccessRule whatchtable= watchTableCmp.Find(watchTable1);
whatchtable.Delete();
```

Error will be thrown in the following cases:

- If the web server has not been activated (WebserverActivate == FALSE), a EngineeringTargetInvocationException is thrown with error details stating that a WatchTableAccessRule can be added only if the web server is enabled
- If a WatchtableAccessRule with the watch table already exists, a ConfigOpenessUserException is thrown with error details stating that the watchtable already exists.
- If you try to add a watch table with WatchAndForceTableAccess.None, a ConfigOpenessUserException with error details stating that WatchAndForceTableAccess.None is not allowed as a access level.

Program code: Assigning force table to the web server

Modify the following program code to search force table in the web server:

```
WatchAndForceTableAccessManager mngr =
deviceItem.GetService<WatchAndForceTableAccessManager>();
ForceTableAccessRuleComposition forceTableCmp = mngr.ForceTableAccessRules;
ForceTableAccess forceTable = forceTableCmp.Find(forceTable1);
```

Modify the following program code to create PLC force table to the web server with read access:

```
ForceTableAccessRuleComposition forceTableCmp = mngr.ForceTableAccessRules;
forceTableCmp.Create(forceTable1, WatchAndForceTableAccess.Read);
```

Watch and force tables at the PLC display

You can use the same Openness service WatchAndForceTableAccessManager available at the Display submodule (DeviceItem) of the PLC DeviceItem. The same web server functionality is used as described above to assign watch and force tables at the PLC display. In contrast to the web server, the display cannot be disabled, so here a similar validation as for WebserverActive is not available.

See also

Export/Import Watch & Force Table (Page 1385)

Connecting to the TIA Portal (Page 82)

5.11.2.15 Managing certificate

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to manage the following tasks with certificates:

- Creating and deleting certificate
- Exporting and importing certificate
- Assigning and unassigning certificates
- Getting certificate ID

Local certificate manager

For device certificates the service LocalCertificateManager at the DeviceItem PLC is provided. The LocalCertificateManager has a local store for the certificates named LocalCertificateStore. There the certificates for the specific PLC are stored. To get the device item that owns the instance of the service, the property OwnedBy can be used.

```
DeviceItem Plc= ...;
// Get local certificate manager
var localCertificateManager = Plc.GetService<LocalCertificateManager>();
// Disable global certificate manager
localCertificateManager.EnableGlobalCertificatesStore = false;
// Get local certificate store
var localCertificateStore = localCertificateManager.LocalCertificateStore;
```

Certificate handling

Modify the following program code to create a certificate:

```
/ Create templates
var templateTls = localCertificateStore.GetCertificateTemplate(CertificateUsage.Tls);
var templateWebserver =
localCertificateStore.GetCertificateTemplate(CertificateUsage.WebServer);
var templateOpcUaServer =
localCertificateStore.GetCertificateTemplate(CertificateUsage.OpcUaServer);//
...
//Template handling and configuration is handled later
// Create certificates
var certificateTls = localCertificateStore.Certificates.Create(templateTls);
var certificateWeb = localCertificateStore.Certificates.Create(templateWebserver);
var certificateOpcUa = localCertificateStore.Certificates.Create(templateOpcUaServer);
```

Modify the following program code to delete and export a certificate:

```
var exportPath = …;
// renew a certificatecertificate
Tls.Delete();
certificateTls = localCertificateStore.Certificates.Create(templateTlsNew);
certificateWeb.Export(new FileInfo(exportPath));
```

Note

With TIA Portal Openness V19, the Export() now accepts exportPath as a parameter and allows you to export certificate in any desired format.

Modify the following program code to assign certificate to web server and OPC UA server using dynamic attributes:

```
DeviceItem opcUaSubmodule = …;
//Find assigned certificates
var foundWebCertificate = Plc.GetAttribute("WebserverCertificate");
var foundOpcUaCertificate = opcUaSubmodule.GetAttribute("OpcUaServerCertificate");
//Assign certificatesPlc.SetAttribute("WebserverCertificate", certificateWeb);
opcUaSubmodule.SetAttribute("OpcUaServerCertificate", certificateOpcUa);
//Unassign certificatesPlc.SetAttribute("WebserverCertificate", null);
opcUaSubmodule.SetAttribute("OpcUaServerCertificate", null);
```

Modify the following program code to import certificates to the local certificate store for a PLC:

```
var certificateWithoutPwd = …;
var certificateWithPwd = …;
var password = new SecureString();
//
…
// Import certificates
// Without password
var importedCertificate1 = certificates.Import(new FileInfo(certificateWithoutPwd));
//With password
var importedCertificate2 = certificates.Import(new FileInfo(certificateWithPwd), password);
```

Modify the following program code to get certificate ID:

```
var certificateId = importedCertificate2.Id;
```

Modify the following program code to indicate if a certificate has a private key the boolean property HasPrivateKey at the certificate object can be used.

```
if(importedCertificate2.HasPrivateKey)
{
//Do something
}
```

Certificate template

The templates are used as basis for creating certificates. New templates can be created at the LocalCertificateStore using the action GetCertificateTemplate(CertificateUsage). Where CertificateUsage is an enumerated with the following possible values:

- None (not supported)
- Tls
- WebServer
- OpcUaServer
- OpcUaClient
- OpcUaClientServer

```
// Create template
var certTemplate = localCertificateStore.GetCertificateTemplate(CertificateUsage.Tls);
```

In a certificate template all properties of a certificate can be set. The access is read-write:

- Signature: The used signature algorithm of type SignatureAlgorithm which is an enumerated with the following possible values
  - None (not supported)
  - Sha1RSA
  - Sha256RSA

```
certTemplate.Signature = SignatureAlgorithm.Sha1RSA;
```

- SubjectAlternativeNames: Is a composition which contains objects of type SubjectAlternativeName (IList of type <SubjectAlternativeName>). With the action Create(SubjectAlternativeNameType, string) a new SAN can be created. The handed over parameters are the Type (of type SubjectAlternativeNameType) and the Value (of type string).
- SubjectAlternativeNameType
  - None (not supported)
  - Dns
  - Email
  - IP
  - Uri

```
var san1 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Dns,
"127.0.0.1");
var san2 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.IP,
"192.168.178.1");
var san3 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Email,
"test@prov.com");
var san4 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Uri,
"something");
san3.Delete();
```

- SubjectCommonName: string

```
string subjectCommonName = certTemplate.SubjectCommonName;
certTemplate.SubjectCommonName = "exampleSubjectCommonName";
```

Usage of type CertificateUsage

```
var usage = certTemplate.Usage;
certTemplate.Usage = CertificateUsage.OpcUaClientServer;
```

ValidFrom: Type DateTime

ValidUntil: Type DateTime

```
var validFrom = certTemplate.ValidFrom;
var validUntilDateTime = new DateTime(2080, 10, 10);
certTemplate.ValidUntil = validUntilDateTime;
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.16 Managing Webserver user of SIWAREX modules

Requirement

- The TIA Portal is connected to the TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to manage the following functionalities of Webserver user of SIWAREX modules:

- Activate or deactivate Webserver user
- Change name of the user
- Setting permission of the user
- Setting password of the user

You can use the SimpleWebserverUserManagement feature to get the list of Webserver user of Siwarex modules. The feature is supported for SIWAREX modules which have Webserver user.

Supported module

Currently only two modules of ET200SP family use this feature:

- 7MH4 138-6CA00-0CU0 V1.0 – TM SIWAREX WP341 HF
- 7MH4 138-6BA00-0CU0 V1.0 – TM SIWAREX WP351 HF

Property

Property name	Data type	Description
UserName	String	To set or get name of the user
Active	Boolean	To add activation of the user
Permissions	SimpleWebserverUserPermissions	To set or get permission of the user. The SimpleWebserverUserPermissions supports the following enums: • SimpleWebserverUserPermissions.ReadWrite • SimpleWebserverUserPermissions.ReadOnly • SimpleWebserverUserPermissions.None (if the user is deactivated) An EngineeringTargetInvocationException is thrown if you set the wrong permission
SetPassword()	Secure String	To set password of the user. The Password can only be set, if the user is active.

Program code

```
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.SimpleWebserverUser;
private void SettingWebserverUser()
{
//To get the list of webserver users
SimpleWebserverUserManagement feature =
subModule.GetService<SimpleWebserverUserManagement>();
// To get the specific user from the list of webserver users
SimpleWebserverUser webserverUser = feature.WebserverUsers[1];
// To set activation of the user
webserverUser.active = true;
// To set the username of the user
webserverUser.UserName = "userName1";
// To get the username of the user
string name = webserverUser.UserName;
// To set the permissions to Read for the user
webserverUser.Permissions = SimpleWebserverUserPermissions.ReadOnly;
SimpleWebserverUserPermissions permission = webserverUser.Permissions;
// To set the password of the user
webserverUser.SetPassword(GetSecureString("Admin123"));
}
```

5.11.2.17 Managing Syslog configuration

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the SysLogConfigurationManager service in TIA Portal Openness to configure enable and disable Syslog feature. The feature enables you the option to select different transport protocol and configure different server for syslog.

The feature is available in S7-1500 PLC firmware version 3.1 onwards.

Property

The SysLogConfigurationManager service provides the following properties to modify the system logging configuration.

Property name	Data type	Description	Access
EnableSystemLogging	Boolean	Enable or Disable the System Logging setting of the SysLog Configuration	Read/Write
TransportProtocol	SysLogTransportProtocolType	Specifies the constants for transport protocol options in system logging configuration. The following enum items and its corresponding value are supported: • None - 0 • TLSServerAndClientAuthentication - 1 • TLSOnlyServerAuthentication - 2 • UDP - 3	Read / Write
SysLogServerConfiguration	SysLogServerConfigurationComposition	Get or set all the server details of the system logging configuration. For details on SysLogServerConfiguration property, refer the below SysLogServerConfiguration section	Read
SysLogAutoAcceptClient	Boolean	Enable or disable the automatic accept certificate at run time setting of the SysLog configuration	Read/Write

Property name	Data type	Description	Access
SysLogClientCertificateId	Uint	Represents client certificate Id for the System logging configuration	Read/Write
SysLogTrustedCertificateIds	String	Represents trusted certificate Id for the System logging configuration	Read/Write

SysLogServerConfiguration property

The SysLogServerConfiguration represents server configuration of the system logging configurations.

- SysLogServerConfiguration Find(string sysLogServerAddress)
- SysLogServerConfiguration Create(string sysLogServerAddress, UInt16 sysLogServerPort)

Property name	Data type	Description
SysLogServerAddress	String	Represents server address of the system logging configuration
SysLogServerPort	UInt16	Represents port number of the system logging configuration

Program code

Modify the following program code to enable or disable the system logging setting of the SysLog Configuration:

```
// To access EnableSysLog Property
var service = plc.GetService<SysLogConfigurationManager>();
service.EnableSystemLogging = true;
```

Modify the following program code to get or set the Transport Protocol of the system logging configuration:

```
var service = plc.GetService<SysLogConfigurationManager>();
service.EnableSystemLogging = true;
service.TransportProtocol = SysLogTransportProtocolType.TLSServerAndClientAuthentication;
```

Modify the following program code to get or set all the server details of the system logging configuration:

```
var service = plc.GetService<SysLogConfigurationManager>();
service.EnableSystemLogging = true;
service.SysLogServerConfiguration.Create("www.test.com", 100);
```

Modify the following program code to enable or disable accept certificate at runtime setting:

```
var service = plc.GetService<SysLogConfigurationManager>();
service.TransportProtocol = SysLogTransportProtocolType.TLSOnlyServerAuthentication;
plc.SetAttribute("SysLogAutoAcceptClient", true);
```

Modify the following program code to represent trusted certificate Id for the system logging configuration:

```
var service = plc.GetService<SysLogConfigurationManager>();
service.EnableSystemLogging = true;
service.TransportProtocol = SysLogTransportProtocolType.TLSServerAndClientAuthentication;
plc.SetAttribute("SysLogTrustedCertificateIds", certificate.Id.ToString());
```

Modify the following program code to represent client certificate Id for the system logging configuration:

```
var service = plc.GetService<SysLogConfigurationManager>();
service.EnableSystemLogging = true;
service.TransportProtocol = SysLogTransportProtocolType.TLSServerAndClientAuthentication;
plc.SetAttribute("SysLogClientCertificateId", (uint)1);
```

5.11.2.18 Supporting secure S7 communication TLS

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to set up the parameters for an online connection.

These online connections are set up on the ConnectionConfiguration object that can be accessed via Configuration navigator of OnlineProvider service. This service exists on a DeviceItem that represents PLC.

```
DeviceItem deviceItem = ...;
OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
ConnectionConfiguration configuration = onlineProvider.Configuration;
```

Depending upon the available connections for a PLC, you can enumerate through the available connection modes or specifiy a certain connection mode. For each mode,you can then specify the PC Interface for that connection type.

The Find-Action of PcInterfaces has two parameters. First the name of the interfaceboard (ConfigurationPcInterface), and additional to this the number of the interfaceboard. Typically the number of interfaceboard is "1".

You can specify the slot for the interface.

```
foreach(ConfigurationMode mode in configuration.Modes)
{
Console.WriteLine("Mode name:{0}", mode.Name);
foreach(ConfigurationPcInterface pcInterface in mode.PcInterfaces)
{
Console.WriteLine("PcInterface name:{0}", pcInterface.Name);
Console.WriteLine("PcInterface number:{0}", pcInterface.Number);
foreach(ConfigurationTargetInterface targetInterface in pcInterface.TargetInterfaces)
{
Console.WriteLine("TargetInterface:{0}", targetInterface.Name);
}
}
}
ConfigurationMode myMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface myPcInterface = myMode.PcInterfaces.Find("PLCSIM", 1);
ConfigurationTargetInterface mySlot = myPcInterface.TargetInterfaces.Find("2 X3");
```

Note

If PLCSIM is not running, it will not be included in the PcInterfaces collection.

You can also choose to set up the connection via the Addresses of a Subnet or a Gateway.

```
foreach(ConfigurationSubnet subnet in pcInterface.Subnets)
{
Console.WriteLine("Subnet name:{0}", subnet.Name);
foreach(ConfigurationAddress subnetAddress in subnet.Addresses)
{
Console.WriteLine("Subnet address:{0}", subnetAddress.Name);
}
foreach(ConfigurationGateway gateway in subnet.Gateways)
{
Console.WriteLine("Gateway name:{0}", gateway.Name);
foreach(ConfigurationAddress gatewayAddress in gateway.Addresses)
{
Console.WriteLine("Gateway address:{0}", gatewayAddress.Name);
}
}
}
```

These settings can than be applied to online connection via the ApplyConfiguration call on the ConnectionConfiguration object. The system will return a boolean value indicating whether the configuration was set successfully.

```
bool result;result = configuration.ApplyConfiguration(slot);
//or
result = configuration.ApplyConfiguration(subnetAddress);
//or
result = configuration.ApplyConfiguration(gatewayAddress);
```

After setting up the connection, you can then go online with the PLC. Each ApplyConfiguration call is independent and any subsequent call will override the previously applied settings. The settings are only valid for the current session, similar to the behavior in the UI. If the settings have already been configured in the UI, then the ApplyConfiguration call is not necessary and the online connection will use the already available settings.

If a device is already online, invoking the ApplyConfiguration will cause an exception to be thrown.

You can determine whether a PLC is already configured by using the IsConfigured property available on the ConnectionConfiguration target:

```
bool isConfigured = configuration.IsConfigured;
```

PLCs that only have one possible configuration will always be configured and therefore, the IsConfigured property will return true without calling ApplyConfiguration.

Note

Configuring the connection and going online will not affect the contents of the offline or online PLCs. The user will have to either download the software and/or hardware to the online PLC or upload the software from the online device in order to make the PLCs consistent.

Legacy communication

From TIA-Portal Version-17, PLCs of type S7-1500 with Firmware (FW) 2.9 communicate vis Tls.

Communication without Tls can be used by the EnableLegacyCommunication.

EnableLegacyCommunication	Description
true	Communication via Tls is enabled
false	false Communication via Tls is disabled. This type of communication is the same as in TIA-Portal Versions < V17.

```
DeviceItem deviceItem = ...;
OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
ConnectionConfiguration configuration = onlineProvider.Configuration;
configuration.EnableLegacyCommunication = true;
```

Note

The security extension is only supported with PLC firmware >=V2.9 (Hardware configuration) from PLC side.

Connection parameters for communicate over Tls

With TIA-Portal V17 PLCs of type S7-1500 are available with Firmware (FW) 2.9. These PLCs support a secure communication over Tls.

The "Transport Layer Security" expects some conditions for a communication. If these condition are not fulfilled, a user decision is necessary whether to trust the communication or not.

For this a event handler can be subscribe on Tls notifications at the ConnectionConfiguration object. If a user decision is necessary, TIA Portal Openness will call this delegate and the client can handle it.

The 'ConnectionConfiguration' object can be accessed via Configuration navigator of OnlineProvider service.

Subscribe to an event of Tls-notification

To subscribe to an event of Tls-notifications you have to register your event handler on OnlineLegitimation:

```
private OnlineConfigurationDelegate m_OnlineConfigurationDelegate = null;
...
ConnectionConfiguration OnlineConfiguration = ...
m_OnlineConfigurationDelegate = OnlineCallBackMethod;
OnlineConfiguration.OnlineLegitimation += m_OnlineConfigurationDelegate;
```

Unsubscribe to an event of Tls-notification

To unsubscribe to an event of Tls-notifications (OnlineLegitimation) you have to register your event handler:

```
ConnectionConfiguration OnlineConfiguration = ...
m_OnlineConfigurationDelegate = OnlineCallBackMethod;
OnlineConfiguration.OnlineLegitimation -= m_OnlineConfigurationDelegate;
```

TlsVerificationConfiguration

When going online to a Plc1500 with a FW 2.9 or higher, a user decision may be necessary as to whether the connection should be established at the given circumstances. In this case the registered event handler delegate is called in the client program. For the given case, the OnlineConfiguration parameter must be checked for "TlsVerificationConfiguration".

The possible values of TlsVerificationConfigurationSelection are:

TlsVerificationConfigurationSelection	Description
NonVerified	The connection is not verified and not trusted
Trusted	The connection is verified and trusted
NonTrusted	The connection is verified and not trusted

```
private void OnlineCallBackMethod(OnlineConfiguration onlineConfiguration)
{
TlsVerificationConfiguration verificationConfiguration = onlineConfiguration as
TlsVerificationConfiguration;
if (verificationConfiguration != null)
{
verificationConfiguration.CurrentSelection =
TlsVerificationConfigurationSelection.Trusted;
//To establish a connection the value Trusted is necessary.
}
}
```

With all other values a go online is not possibly and leads to an user exception.

Also TlsVerificationConfiguration supports two attributes:

Name of method	Description
PlcName	The name of the PLC calling the delegate.
VerificationInfo	Message for the user. Contains reason for calling the delegate.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.19 Updating module description

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

In TIA Portal Openness, a specific service ModuleDescriptionUpdater is used at the device item to update the current module description to the newest version of the module. To get the device item that owns the instance of the service, the property OwnedBy can be used.

You can use the CanUpdate attribute to show if a new ConfigObject version for this deviceitem is available.

Atribute	Data type	Description
CanUpdate	bool	TRUE: A new version is available FALSE: There is no new version

You can use the action UpdateModuleDescription ( ) to update the ConfigObject version of a deviceitem.

Action	Return value	Description
UpdateModuleDescription	True	The deviceitem is up to date: • The deviceitem was updated successfully • The deviceitem was already up to date
	False	The deviceitem is not up to date: • The deviceitem could not be updated

Program code

```
DeviceItem deviceItem = ...;
var descriptionUpdater = deviceItem.GetService<ModuleDescriptionUpdater>();
if (descriptionUpdater != null)
{
if (descriptionUpdater.CanUpdate) //e.g is update module version possible
{
bool result = descriptionUpdater.UpdateModuleDescription();
.
.
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.20 Supporting UMAC on PLC device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to perform actions on PLC device protected via User Management Access Control (UMAC).

A legitimation via the event handler OnlineLegitimation is required for all online PLC functions that require legitimation and which support UMAC. This is the same event handler that is used for communication via TLS. For information about OnlineLegitimation event

handler, see Subscribe to an event of Tls-notification section of Supporting secure S7 communication TLS (Page 475)

To register the event handler on online legitimation,you need the connection configuration object. You can access the Connection Configuration object via Configuration navigator of the corresponding online service (e.g. OnlineProvider, FingerprintDataProvider, DownloadProvider, StationUploadProvider, etc.).

Program code: Accessing connection configuration object

To access the configuration object modify the following program code:

```
...
DeviceItem deviceItem = ...;
DownloadProvider downloadProvider = deviceItem.GetService<DownloadProvider>();
ConnectionConfiguration onlineConfiguration = downloadProvider.Configuration;
...
```

Program code: Subscribe to an event for online legitimation

To subscribe to an event of online legitimation notifications you have to register your event handler on OnlineLegitimation:

```
ConnectionConfiguration onlineConfiguration = ...;
OnlineConfigurationDelegate onlineConfigurationDelegate = OnlineCallBackMethod;
onlineConfiguration.OnlineLegitimation += onlineConfigurationDelegate;
...
private void OnlineCallBackMethod(OnlineConfiguration onlineConfiguration)
{
...
```

Note

Registering for this event is only necessary if it has not been done in the context of Tls-communication.

Program code: Unsubscribe to an event for online legitimation

To unsubscribe to an event of online notifications (OnlineLegitimation) you have to unregister your event handler:

```
ConnectionConfiguration onlineConfiguration = ...;
OnlineConfigurationDelegate onlineConfigurationDelegate = OnlineCallBackMethod;
onlineConfiguration.OnlineLegitimation -= onlineConfigurationDelegate;
...
private void OnlineCallBackMethod(OnlineConfiguration onlineConfiguration)
{
..
```

OnlineAuthenticationConfiguration

The OnlineAuthenticationConfiguration has the following properties and methods:

OnlineAuthenticationConfiguration	Description
bool IsSecureCommunication { get; }	The property provides 'true' in the case of a "Secure communication" (TLS handshake) connection to the PLC. In all other cases the attribute delivers 'false'. For more information, see Downloading to PLC devices (Page 385)
IList<AuthenticationType> GetSupportedAuthenticationTypes()	Get all supported AuthenticationTypes via GetSupportedAuthenticationTypes() from the PLC.
OnlineCredentials OnlineCredentials { get; }	Property to configure the credential details.

IsSecureCommunication

In the OnlineAuthenticationConfiguration object, You can verify whether the connection is using secure Tls by using the IsSecureCommunication option. This is similar to the configuration for upload or download through the DownloadConfiguration.

```
OnlineAuthenticationConfiguration onlineAuthenticationConfiguration = onlineConfiguration
as OnlineAuthenticationConfiguration;
if (onlineAuthenticationConfiguration != null)
{
    // Secure communication via TLS?
    if (onlineAuthenticationConfiguration.IsSecureCommunication)
    {
      ...
    }
}
```

GetSupportedAuthenticationTypes() and AuthenticationType

GetSupportedAuthenticationTypes()

The different authentication legitimation types are possible depending on the downloaded UMAC configuration. To get the supported types you have to call:

```
OnlineAuthenticationConfiguration onlineAuthenticationConfiguration = onlineConfiguration
as OnlineAuthenticationConfiguration;
if (onlineAuthenticationConfiguration != null)
{
    IList<AuthenticationType> supportedTypes =
onlineAuthenticationConfiguration.GetSupportedAuthenticationTypes();
    foreach (AuthenticationType supportedType in supportedTypes)
    {
       ...
    }
}
```

AuthenticationType

The AuthenticationType represents the type of authentication that you can choose.
To get the type of authentication, you can get the UserType value of the UserType at the CurrentUserType attribute.

AuthenticationType	Attribute	Description
UserType CurrentUserType { get; }	Deliver the UserType, supported by the PLC	This value is usable in OnlineCredentials: UserType The possible UserType entries for Legitimation supported by the PLC are: • None - No legitimation is required • AnonymousUser - Legitimation as an anonymous user • GlobalUser - Legitimation as a global user • ProjectUser - Legitimation as a project user • SingleSignOnUser - Single sign-on user • PasswordOnly - Legacy legitimation with PLC protection access level password only

```
...
IList<AuthenticationType> supportedTypes =
onlineAuthenticationConfiguration.GetSupportedAuthenticationTypes();
foreach (AuthenticationType supportedType in supportedTypes)
{
    UserType userType = supportedType.CurrentUserType;
    if (userType == UserType.ProjectUser)
    {
       ...
    }
}
...
```

OnlineCredentials

The OnlineCredentials has the following properties and methods:

OnlineCredentials	Action	Description
void SetPassword(SecureString password)	Set the password via SetPassword method.	Enter a password into the OnlineCredentials to gain access to the module.
string Name { get; set; }	Get or set the UserName via Name property.	Enter name as UserName into the OnlineCredentials.
UserType Type { get; set; }	Get or set the UserType via Type poperty	Enter UserType into the OnlineCredentials. The possibly values are available via GetSupportedAuthenticationTypes()

After you verify the supported user types and necessary legitimation informations, you have to set these within the OnlineCredentials:

```
OnlineAuthenticationConfiguration onlineAuthenticationConfiguration = onlineConfiguration
as OnlineAuthenticationConfiguration;
if (onlineAuthenticationConfiguration != null)
{
   // Check supported user types and legitimation information
   ...
   // Set the online credentials for legitimation
   OnlineCredentials onlineCredentials =
onlineAuthenticationConfiguration.OnlineCredentials;
   SecureString password = ...;
   onlineCredentials.Type = UserType.ProjectUser;
   onlineCredentials.Name = "User1";
   onlineCredentials.SetPassword(password);
}
```

Note

The explicit legitimation is not required within the callback. The legitimation will be done after the callback return. In case of insufficient access rights, the online action (e.g. download) will abort with an exception of an illegitimate connection.

5.11.2.21 Supporting IP Accessibility

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 128)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to support IP accessibility. In TIA Portal Openness, a new dynamic attribute PlcAccessCommunicationModule gets available at the device item CPU. The attribute is of type object, so that you can access to the CP object you want to select for IP Accessibility and assign the object to PlcAccessCommunicationModule.

If a CP has already been selected you can retrieve the CP object by getting the value of PlcAccessCommunicationModule.

Program code

```
DeviceItem Plc = ...;
object communicationModule = Plc.GetAttribute("PlcAccessCommunicationModule");
// cP contains the object reference of a plugged CP
Plc.SetAttribute("PlcAccessCommunicationModule", cP);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.22 Accessing PLC access control configuration provider

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the PlcAccessControlConfigurationProvider service in TIA Portal Openness to enable and disable the access control configuration at run time.

Method, Parameter, and Enum

Method

The PlcAccessControlConfigurationProvider service provides an attribute called PlcAccessControlConfiguration, which supports the following functions to configure the access control configuration.

Method name	Return Datatype	Description
GetAttribute(string attributeName)	integer	Used to get PlcAccessControlConfiguration
SetAttribute(string attributeName, int configuration)	void	Used to set PlcAccessControlConfiguration

Note

All S7-1500 PLCs having firmware version 3.1 and above supports PlcAccessControlConfiguration.

Parameter

The GetAttribute() method supports the following parameter to get PlcAccess ControlConfiguration:

Parameter name	Data type	Description
attributeName	string	Specifies the name of the parameter, for example "PlcAccessControlConfiguration" in this case. The possible return Enum items are: • Disabled • EnabledWithAccessControl • EnabledWithAccessControlViaAccessLevel

The SetAttribute() method supports the following parameter to set PlcAccess ControlConfiguration:

Parameter name	Data type	Description
attributeName	string	Specifies the name of the parameter, for example "PlcAccess-ControlConfiguration" in this case
configuration	The possible return Enum items are: • Disabled • EnabledWithAccessControl • EnabledWithAccessCon-trolViaAccessLevel	Valid Plc Access Control Configuration. It returns void for invalid scenarios, which will be handled by the exception.

Enum

The PlcAccessControlConfiguration provides the following Enum values and its description:

Enum	Value	Description
Disabled	0	No change in accessing the PLC through Automated System.
EnabledWithAccessControl	2	There won't be separate configuration for different modes of accessing the PLC like (AccessLevel,WebServer,OPCUA).
EnabledWithAccessControl-ViaAccessLevel	3	There won't be separate configuration for different modes of accessing the PLC like (WebServer,OPCUA) however you can access the PLC through access level passwords.

Program code: GetAttribute()

```
// Example to use 'GetAttribute'
PlcAccessControlConfigurationProvider provider =
plc.GetService<PlcAccessControlConfigurationProvider>();
var configuration = provider.GetAttribute("PlcAccessControlConfiguration");
```

Program code: SetAttribute()

```
// Example to use 'SetAttribute'
PlcAccessControlConfigurationProvider provider =
plc.GetService<PlcAccessControlConfigurationProvider>();
provider.SetAttribute("PlcAccessControlConfiguration",
PlcAccessControlConfiguration.EnabledWithAccessControlViaAccessLevel);
```

5.11.2.23 Setting a display password

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal Openness to set the display protection password. You can use the method SetPassword(SecureString password) to set the display protection password where the parameter for the password to be a SecureString. The confirmation password is not needed in Openness and therefore not available.

The feature will be available if the following preconditions are met.

- The PLC has a "Display" instance.
- The "Display" supports "Password" protection feature. There are cases eg. Software PLC where Display is available but no password protection available.

Program code

Modify the following program code to get the device item that owns the instance of the service, the property OwnedBy can be used.

```
DisplayProtection  displayProtection  = plcDisplay.GetService<DisplayProtection>();
var displayDeviceItem = displayProtection.OwnedBy;
```

Modify the following program code to set the Display Protection password:

```
displayProtection.SetPassword(someSecuredString);
//Sets the string someSecuredString as the CPU display protection password.
```

Note

SW controllers have a display submodule, but do not support any display protection. So in TIA Portal Openness the feature SetPassword is not available at the display submodule of any SW controllers.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.2.24 Accessing web server and OPC UA user management

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal Openness to access web server and OPC UA submodule of PLCs. It is possible to add a user up to maximum number at the web server and the OPC UA submodule of PLCs. The user has a user name and a password.

User management at Web Server

You can perform the following operations, such as adding user, deleting user, and setting password at web server for all supported device only if web server is activated at the module.

If the web sever is not activated, the add, delete and set operations will throw a EngineeringTargetInvocationException, but still the read operations will be available.

The WebServerUserManagement service will be available for the PLC instance of only the supported devices. For other non-supported devices, the service will be null.

```
WebServerUserManagement
{
Navigators WebServerUsers
{
WebServerUserComposition
}
}
```

The Service provides a navigator called WebServerUsers using which a WebServerUserComposition can be obtained, which manages the WebServerUser instances.

```
WebServerUserComposition
{
WebServerUser Find(string username);
void Create(string username, WebserverUserPermissions permissions,SecureString password);
}
WebServerUser
{
string UserName{get;}
WebserverUserPermissions Permissions {get;set;}
void Delete();
void SetPassword(SecureString password);
}
```

Program code: Actions at WebServerUserComposition

Modify the following program to search web server user:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
```

Modify the following program code to create a new web server user with a web server user permissions:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Create("user1",
WebserverUserPermissions.ReadTagStatus, someSecureString);
```

Modify the following program code to delete the web server user:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
user1.Delete();
```

Modify the following program code to replace the current password of the user with the new password:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
user1.SetPassword(someSecureString);
```

User Management at OPC UA

You can perform the perform following operations such as adding user, deleting user, and setting password at the OPC UA Server only if the OPC UA Server is activated and username and password authentication is enabled.

If these conditions are not met, the operation throws a EngineeringTargetInvocationException with error details stating that the authentication needs to be enabled.

The OpcUaUserManagement service will be available on the OPC UA submodule of the PLC. The Service will be available for all supported devices where OPC UA submodule is available for the device item. In other cases, the service will be returned as null.

```
OpcUaUserManagement
{
Navigators OpcUaUsers
{
OpcUaUserComposition
}
}
```

The Service provides a navigator called OpcUaUsers using which a OpcUaUserComposition can be obtained. The composition manages the OpcUaUser instances**:**

```
OpcUaUserComposition
{
OpcUaUser Find(string username);
void Create(string username, SecureString password);
}
OpcUaUser
{
string UserName{get;}
void Delete();
void SetPassword(SecureString password);
}
```

Program code: Actions at OpcUaUserComposition

Modify the following program code to search a OpcUaUser:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user1 = opcUaUserComposition.Find("user1");
```

Modify the following program code to create a new OpcUaUser:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
opcUaUserComposition.Create("user1", someSecureString);
```

Modify the following program code to delete the OpcUaUser:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user = opcUaUserComposition.Find("user1");
user.Delete();
```

Modify and use the program code to replace the current password with a supplied one:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user = opcUaUserComposition.Find("user1");
user.SetPassword(someSecureString);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.3 Blocks

5.11.3.1 Querying the "Program blocks" group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.

Program code

Modify the following program code to query the group "Program blocks":

```
private static void GetBlockGroupOfPLC(PlcSoftware plcsoftware)
//Retrieves the system group of a block
{
    PlcBlockSystemGroup blockGroup = plcsoftware.BlockGroup;
}
```

5.11.3.2 Querying the system group for system blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code:

Modify the following program code to determine the group created for system blocks by the system:

```
PlcSoftware plcSoftware = ...
foreach (PlcSystemBlockGroup systemGroup in plcSoftware.BlockGroup.SystemBlockGroups)
{
    foreach (PlcSystemBlockGroup group in systemGroup.Groups)
    {
        PlcBlockComposition pbComposition = group.Blocks;
        foreach (PlcBlock block in pbComposition)
        {
            //add your code here
        }
    }
}
```

5.11.3.3 Enumerating system subgroups

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code: Enumerating all system subgroups

Modify the following program code to enumerate the system subgroups of all system blocks:

```
//Retrieves the system generated group for system blocks
private static void GetSystemgroupForSystemblocks(PlcSoftware plcSoftware)
{
    PlcSystemBlockGroupComposition systemBlockGroups =
plcSoftware.BlockGroup.SystemBlockGroups;
    if (systemBlockGroups.Count != 0)
    {
        PlcSystemBlockGroup sbSystemGroup = systemBlockGroups[0];
        foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
        {
            EnumerateSystemBlockGroups(group);
        }
    }
}
private static void EnumerateSystemBlockGroups(PlcSystemBlockGroup systemBlockGroup)
{
    foreach (PlcSystemBlockGroup group in systemBlockGroup.Groups)
    {
        // recursion EnumerateSystemBlockGroups(group);
    }
}
```

Program code: Accessing a specific subgroup

Modify the following program code to access a specific subgroup:

```
private static void AccessSbGroup(PlcSystemBlockGroup systemBlockGroup)
{
    PlcSystemBlockGroup group1 = systemBlockGroup.Groups.Find("User group XYZ");
    PlcSystemBlockGroup group2 = group1.Groups.Find("User group ZYX");
}
```

See also

Adding an external file (Page 509)

5.11.3.4 Enumerating user-defined block groups

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.

Application

Subgroups are taken into account recursively for enumeration.

Program code: Enumerating all groups

Modify the following program code to enumerate the user-defined block groups:

```
//Enumerates all block user groups including sub groups
private static void EnumerateAllBlockGroupsAndSubgroups(PlcSoftware plcsoftware)
{
    foreach (PlcBlockUserGroup blockUserGroup in plcsoftware.BlockGroup.Groups)
    {
        EnumerateBlockUserGroups(blockUserGroup);
    }
}

private static void EnumerateBlockUserGroups(PlcBlockUserGroup blockUserGroup)
{
    foreach (PlcBlockUserGroup subBlockUserGroup in blockUserGroup.Groups)
    {
        EnumerateBlockUserGroups(subBlockUserGroup);
        // recursion
    }
}
```

Program code: Accessing a group

Modify the following program code to access a selected user-defined block group:

```
//Gives individual access to a specific block user group
private static void AccessBlockusergroup(PlcSoftware plcsoftware)
{
    PlcBlockUserGroupComposition userGroupComposition = plcsoftware.BlockGroup.Groups;
    PlcBlockUserGroup plcBlockUserGroup = userGroupComposition.Find("MyUserfolder");
}
```

5.11.3.5 Enumerating all blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.

Application

Targeted access to a program block is possible if its name is known.

Program code: Enumerating all blocks

Modify the following program code to enumerate the blocks of all block groups:

```
private static void EnumerateAllBlocks(PlcSoftware plcsoftware)
//Enumerates all blocks
{
    foreach (PlcBlock block in plcsoftware.BlockGroup.Blocks)
    {
        // Do something...
    }
}
```

Program code: Accessing a specific block

Modify the following program code to access a specific block:

```
private static void AccessASingleBlock(PlcSoftware plcsoftware)
//Gives individual access to a block
{
    // The parameter specifies the name of the block
    PlcBlock block = plcsoftware.BlockGroup.Blocks.Find("MyBlock");
}
```

5.11.3.6 Querying information of a block/user data type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The TIA Portal Openness API supports the querying of the following information for program and data blocks and for user data types:

- Time stamp in UTC time format.
  You check the following with the time stamp:
  - When the block was last compiled.
  - When the block was last changed.
- "Consistency" attribute
  The "Consistency" attribute is set to "True" in the following cases:
  - The block has been successfully compiled.
  - The block has not been changed since compilation.
  - No changes that would require re-compilation have been made to external objects.
- Programming language used (program and data blocks only)
- Block number
- Block name
- Block author
- Block family
- Block title
- Block version

See also Blocks and types of the TIA Portal Openness object model (Page 59) for further information.

Program code

Modify the following program code to query the information listed above:

```
private static void GetPlcBlockInformation(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    // Read information
    DateTime compileDate = plcBlock.CompileDate;
    DateTime modifiedDate = plcBlock.ModifiedDate;
    bool isConsistent = plcBlock.IsConsistent;
    int blockNumber = plcBlock.Number;
    string blockName = plcBlock.Name;
    ProgrammingLanguage programmingLanguage = plcBlock.ProgrammingLanguage;
    string blockAuthor = plcBlock.HeaderAuthor;
    string blockFamily = plcBlock.HeaderFamily;
    string blockTitle = plcBlock.HeaderName;
    System.Version blockVersion = plcBlock.HeaderVersion;
}
```

See also

Importing configuration data (Page 1212)

5.11.3.7 Setting and removing protections from a block

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Application

You can use the TIA Portal Openness to set or remove the password protection of a block via the PlcBlockProtectionProvider class and the PlcBlockProtectionProvider service. The service PlcBlockProtectionProvider is accessible on blocks which fulfill the following conditions:

- Block is know-how protectable
- Block is a code block or a global DB
- Block must be compiled
- Block is supported or editable in the current PLC
- Block is not in readonly context
- Block is not know-how protected
- Block is not online
- Block is not a CPU-DB
- Block is not of classic encryption language, ProDiag or ProDiag-OB
- Block is not an encrypted imported classic block

In case the block doesn't fulfill all conditions a null reference is returned by GetService() method.

Protect a block

Use the Protect() method to set the password to protect the porgramming block with.

```
void Protect(SecureString password)
```

Errors will occur for following cases:

- An attempt to protect an already protected block: An EngineeringTargetInvocationException will be thrown with the message "You can't protect an already protected object".
- An attempt to protect with an empty string as password: An EngineeringTargetInvocationException will be thrown with the message "Password was not specified".
- An attempt to protect a failsafe block when the failsafe-program is password protected: An EngineeringTargetInvocationException will be thrown.
- An attempt to protect a failsafe block when the block is not called: An EngineeringTargetInvocationException will be thrown.

Unprotect a block

Use the Unprotect() method to remove the password the porgramming block is protected with.

```
void Unprotect(SecureString password)
```

Errors will occur in the following cases:

- An attempt to unprotect an already unprotected block: An EngineeringTargetInvocationException will be thrown with the message "You can't unprotect an object without protection".
- An attempt to unprotect with wrong password: An EngineeringTargetInvocationException will be thrown with the message "The used password was refused".
- An attempt to protect with an empty string as password: An EngineeringTargetInvocationException will be thrown with the message "Password was not specified".

Program code

```
PlcBlock block = ...;
PlcBlockProtectionProvider protectionProvider =
block.GetService<PlcBlockProtectionProvider>();
if (protectionProvider != null)
{
// ...
// perform know-how protection related operations here
}
```

Check for invalid characters

You can use any characters including backspace, tab etc. to protect a block with the Protect() method. The password submitted to protect a block the password is a SecureString. You can use the GetInvalidPasswordCharacters() to check if the provided password has invalid characters.

```
SecureString CreatePasswordString(ProtectionProvider protectionProvider, IEnumerable<char>
contentCharacters)
{
    IList<char> invalidCharacters = protectionProvider.GetInvalidPasswordCharacters();
    SecureString password = new SecureString();
    foreach(char ch in contentCharacters)
    {
        if (!invalidCharacters.Contains(ch))
        {
            password.AppendChar(ch);
        }
        else
        {
            // at least one of the content characters is not valid
            // signal an error - e.g. throw an exception
            ...
        }
    }
    return password;
}
```

Errors will occur in the following cases:

- An attempt to unprotect an already unprotected block: An EngineeringTargetInvocationException will be thrown with the message "You can't unprotect an object without protection".
- An attempt to unprotect with wrong password: An EngineeringTargetInvocationException will be thrown with the message "The used password was refused".
- An attempt to protect with an empty string as password: An EngineeringTargetInvocationException will be thrown with the message "Password was not specified".

5.11.3.8 Deleting block

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Program code

Modify the following program code to delete a block:

```
//Runs through block group and deletes blocks
private static void DeleteBlocks(PlcSoftware plcsoftware)
{
    PlcBlockSystemGroup group = plcsoftware.BlockGroup;
    //  or BlockUserGroup group = ...;
    for (int i = group.Blocks.Count - 1; i >= 0; i--)
    {
        PlcBlock block = group.Blocks[i];
        if (block != null)
        {
            block.Delete();
        }
    }
}
```

See also

Importing configuration data (Page 1212)

5.11.3.9 Creating group for blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to create a group for blocks:

```
private static void CreateBlockGroup(PlcSoftware plcsoftware)
//Creates a block group
{
    PlcBlockSystemGroup systemGroup = plcsoftware.BlockGroup;
    PlcBlockUserGroupComposition groupComposition = systemGroup.Groups;
    PlcBlockUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
}
```

See also

Importing configuration data (Page 1212)

5.11.3.10 Deleting group for blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Program code

Modify the following program code to delete a group for blocks:

```
// Deletes user groups from PlcBlockSystemGroup or PlcBlockUserGroup
private static void DeleteBlockFolder(PlcSoftware plcSoftware)
{
    PlcBlockUserGroup group = plcSoftware.BlockGroup.Groups.Find("myGroup");
    //PlcBlockSystemGroup group = plcSoftware.BlockGroup;
    PlcBlockUserGroupComposition subgroups = group.Groups;
    PlcBlockUserGroup subgroup = subgroups.Find("myUserGroup");
    if (subgroup != null)
    {
        subgroup.Delete();
    }
}
```

See also

Importing configuration data (Page 1212)

5.11.3.11 Accessing attributes of all blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to set the attributes applicable for all blocks using SetAttribute() and SetAttributes() method.

The SetAttributes() can operate on all objects types where writtable openness attribute is available. With one input item it behaves as SetAttribute.

Parameter checking:

- Initial checks are done before the first item is being processed by global PE infrastructure: checking of input items for identical and valid attribute's name, data type mismatch. The same attribute cannot be set twice by one SetAttributes
- Sequential check happens during running of SetAttributes. The order of the processing is the same as the order of the input list elements

The SetAttributes operates in only one transaction. If any attribute of input list cannot be set, the values of attributes are remaining unchanged. SetAttributes sets attributes in the same order as they can be found in the parameter list. The parameters can be dependent or independent of each other. If one parameter depends on one of the previous parameters in the list, this parameter is checked and evaluated according to the current value of the other attribute (not to the original value at the time of calling). If the SetAttributes cannot be executed, details of the error are indicated. The error message contains the first attribute name and its value which could not be managed to set and its reason.

The following program code examples were given based on the two attributes AutoNumber and Number using SetAttribute() (Refer Exporting blocks (Page 1351) for all applicable attributes of blocks).

Program code: SetAttribute

```
...
PlcBlockGroup blockFolder = YourUtilities.GetFolder();
var block = blockFolder.Blocks.Find("Block_1");
 if ((bool)block.GetAttribute("AutoNumber")==true)
   {
      block.SetAttribute("AutoNumber",false);
   }
 block.SetAttribute("Number",2);
...
```

Program code: SetAttributes

```
PlcBlock block = SelectBlock("MC-Servo");
if (block != null)
{
IList<KeyValuePair<string, object>> list = new List<KeyValuePair<string, object>>()
{
new KeyValuePair<string, object>("DataExchangeMode", OBDataExchangeMode.Synchronous),new
KeyValuePair<string, object>("SynchronousApplicationCycleTime", (float)69)
};
 try
  {
     block.SetAttributes(list);
  }
 catch (EngineeringException e)
  {
    Console.WriteLine("Exception: " + e.Message);
  }
}
```

5.11.3.12 Creating a ProDiag-FB

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the PLCBlock composition's create action to create ProDiag FB with the following parameters.

1. Name
2. Auto number flag
3. Number
   - If "auto number flag" is true, the given block number will be set in case it is free, otherwise a new block will be generated
   - If "auto number flag" is false, the given block number will be set to the block
4. Programming language
   - Invoke create action with ProDiag programming language, then a new FB will be created without IDB.
   - Invoke create action with IDB of ProDiag, than IDB of ProDiag will be created.
   - In any other not supported case, a recoverable exception is thrown.

Program code: Creating a ProDiag-FB

```
PlcSoftware plc = ...;
PlcBlockGroup blockFolder = plc.BlockGroup;
PlcBlockComposition blockComposition = blockFolder.Blocks;
if (blockComposition != null)
{
string fbName = "ProDiag_Block";
bool isAutoNumber = true;
int number = 1;
var progLang = ProgrammingLanguage.ProDiag;
FB block = blockComposition.CreateFB(fbName, isAutoNumber, number, progLang);
string iDBName="ProDiag_IDB";
string instanceOfName = fbName;
InstanceDB iDbBlock = blockComposition.CreateInstanceDB(iDBName, isAutoNumber, number,
instanceOfName);
```

See also

Accessing supervisions and properties of ProDiag-FB (Page 504)

5.11.3.13 Accessing supervisions and properties of ProDiag-FB

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Accessing supervisions of User-FB

You can use the TIA Portal Openness to access the supervisions at FB block. Every FB block has the list of supervisions including Classic and Plus PLCs. You can set AssignedProDiagFB at

InstanceDB via the attribute AssignedProDiagFB (Refer Exporting blocks (Page 1351)).

You can use the GetAttribute(), GetAttributes(), and SetAttribute() method for accessing the attributes.

Restriction

You cannot use the SetAttributes() in the TIA Portal Openness to set more than one attribute. You will get an exception in TIA Portal Openness for trying to set more than one attribute using SetAttributes().

Program code: Accessing supervisions of ProDiag-FB

```
…
PlcBlock iDB = plc.BlockGroup.Blocks.Find("FB_Block_DB");
string fbName = iDB.GetAttribute("InstanceOfName").ToString();
FB fb = (FB)plc.BlockGroup.Blocks.Find(fbName);
if (fb.Supervisions.Count > 0)Console.WriteLine("Contains supervisions");
else
Console.WriteLine("Does not contains supervisions");
…
```

Program code: Getting and setting the assigned ProDiag-FB at and IDB

```
...
 PlcBlockGroup blockFolder = plc.BlockGroup;
 PlcBlock instanceDB = blockFolder.Blocks.Find("IDB");
 PlcBlock plcProdiag = blockFolder.Blocks.Find("block_Prodiag");
 instanceDB.SetAttribute("AssignedProDiagFB", plcProdiag.Name);
 var assignedProDiagFB = instanceDB.GetAttribute("AssignedProDiagFB");
...
```

See also

Creating a ProDiag-FB (Page 503)

5.11.3.14 Assigning ProDiag FB for DBs & Tag Table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal Openness to get and set the assigned proDiagFB value of a member in Global DB and TagTable using GetAttribute/SetAttribute method.

The Siemens.Engineering.SW.Blocks.Interface.Member type name provides the following properties:

Property Name	Data type	Access
Name	String	Read
AssignedProDiagFB	String	Read/Write

Program code: Getting proDiagFB Value from DB member & Tag member

Modify the following program code to get prodiagFB in different instances of UDT irrespective of whether the UDT has supervision or not:

```
//Simple UDT instance
MemberComposition gdbMembers = dataBlock.Interface.Members;
Member member = gdbMembers.Find("SimpleUdtInstanceMember");
object proDiagFB = member.GetAttribute("AssignedProDiagFB");
//Array of UDT Instance
Member member = dataBlock.Interface.Members.Find("ArrayUdtMember.ArrayUdtMember[0]");
object proDiagFB = member.GetAttribute("AssignedProDiagFB");
//UDT instance inside Struct
Member member = dataBlock.Interface.Members.Find("structMember.UdtInstanceMember");
object proDiagFB = member.GetAttribute("AssignedProDiagFB");
//UDT Instance inside Array of Struct
MemberComposition gdbMembers = dataBlock.Interface.Members;
Member member =
gdbMembers.Find("ArrayOfStructMember.ArrayOfStructMember[0].UdtInstanceMember");
object proDiagFB = member.GetAttribute("AssignedProDiagFB");
```

Modify the following program code to get prodiagFB in tagTable:

```
// UDT Instance in TagTable
PlcTagTable tagTable = plc.TagTableGroup.TagTables.Find("Tag table_1");
PlcTag tag = tagTable.Tags.Find("Tag_1");
object proDiagFB = tag.GetAttribute("AssignedProDiagFB ");
```

Program code: Setting proDiagFB Value to DB & Tag member

Modify the following program code to set assigned ProdiagFB attribute to a member in DB:

```
//Simple UDT instance
MemberComposition gdbMembers = dataBlock.Interface.Members;
Member member = dataBlock.Interface.Members.Find("SimpleUdtInstanceMember");
member.SetAttribute("AssignedProDiagFB", "ProDiagFBName");
//Array of UDT Instance
Member member = dataBlock.Interface.Members.Find("ArrayUdtMember.ArrayUdtMember[0]");
member.SetAttribute("AssignedProDiagFB", "ProDiagFBName");
//UDT instance inside Struct
Member member = dataBlock.Interface.Members.Find("structMember.UdtInstanceMember");
member.SetAttribute("AssignedProDiagFB", "ProDiagFBName");
//UDT Instance inside Array of Struct
Member member =
dataBlock.Interface.Members.Find("ArrayOfStructMember.ArrayOfStructMember[0].UdtInstanceMe
mber");
member.SetAttribute("AssignedProDiagFB", "ProDiagFBName");
```

Modify the following program code to set the assigned ProdiagFB attribute to a tag member:

```
// UDT Instance in TagTable
var tagTable = plc.TagTableGroup.TagTables.Find("Tag table_1");
var tag = tagTable.Tags.Find("Tag_1");
tag.SetAttribute ("AssignedProDiagFB ", "ProDiagFBName");
```

Note

The GetAttribute and SetAttribute of ProDiagFB is a dynamic attribute and valid only for UDT instance in Global Db and Tag Table.

Objects like simple datatype for example int, string etc and derived datatype like struct, array, all system defined types , the user constants, and system constants types do not support this attribute.

Exception will be thrown if you try to get/set the attribute for an object which are not valid.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.3.15 Reading ProDiag-FB blocks and attributes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project via a TIA Portal Openness application
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to read the ProDiag function block version, and other ProDiag related attribute values. You can use GetAttribute ( ) and GetAttributes ( ) methods to read the ProDiag FBs language specific attributes present.

Attributes

The following attributes are supported by ProDiag-FB in Openness:

Attributes	Type
ProDiagVersion	Version
InitialValueAcquisition	bool
UseCentralTimeStamp	bool

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.3.16 Exporting ProDiag alarm message

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the ExportProDIAGInfo() on CodeBlock in the TIA Portal Openness to export alarm message for a specific ProDiag block in csv format. The resulting output can be opened in

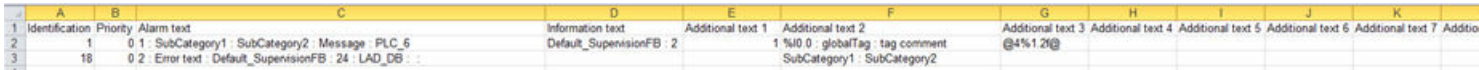
microsoft excel.

To export the ProDiag block in csv format, you make sure that the programming language of a ProDiag-FB is "ProDiag". Otherwise, a recoverable exception will be thrown to you.

Export file

The content of the ProDiag exported file contains the identification consisting of the client alarm ID and the supervision ID in hexadecimal format. The second column includes the priority of the supervision category.

An example for the content of an exported file is shown below.



	A	B	C	D	E	F	G	H	I	J	K	
1	Identification	Priority	Alarm text	Information text	Additional text 1	Additional text 2	Additional text 3	Additional text 4	Additional text 5	Additional text 6	Additional text 7	Additio
2	1	0	1 : SubCategory1 : SubCategory2 : Message : PLC_6	Default_SupervisionFB : 2	1	%I0.0 : globalTag : tag comment	@4%1.2f@					
3	18	0	2 : Error text : Default_SupervisionFB : 24 : LAD_DB : :			SubCategory1 : SubCategory2						
4												

Program code

```
private static void ExportProDiagBlock(PlcBlock plcBlock)
{
FB functionBlock = plcBlock as FB;
if (functionBlock != null && functionBlock.ProgrammingLanguage ==
ProgrammingLanguage.ProDiag && functionBlock.IsConsistent)
{
// Define absolute directory path where the CSV files will be stored
string directoryPath = Path.Combine(@"D:\Temp\ProDiagInfos", functionBlock.Name);
DirectoryInfo directoryInfo = new DirectoryInfo(directoryPath);
// Export the alarm messages of a ProDiag block in CSV format
functionBlock.ExportProDIAGInfo(directoryInfo);
}
}
```

5.11.3.17 Adding an external file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project via a TIA Portal Openness application:
  See Opening a project (Page 128)

Application

You can add an external file to a PLC. This external file is stored in the file system under the defined path.

The following formats are supported:

- STL
- SCL
- DB
- UDT

Note

Accessing groups in the "External source files" folder is not supported.

An exception is thrown if you specify a file extension other than *.AWL, *.SCL, *.DB or *.UDT.

Program code

Modify the following program code to create an external file in the "External source files" folder from a block.

```
private static void CreateBlockFromFile(PlcSoftware plcSoftware)
// Creates a block from a AWL, SCL, DB or UDT file
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.CreateFromFile("SomeBlockNameHere","SomePa
thHere");
}
```

5.11.3.18 Generate source from block

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Introduction

The TIA Portal Openness API interface supports the generation of sources in UTF-8 from STL or SCL blocks, data blocks and PLCTypes (user data types). To generate a source file of a block, invoke the method `GenerateSource` on the `PlcExternalSourceSystemGroup` instance.

The scale of the generated source file depends on the generation option of this function:

- GenerateOptions.None
  Generate source from provided blocks only.
- GenerateOptions.WithDependencies
  Generate source including all dependent objects.

The interface `Siemens.Engineering.SW.ExternalSources.IGenerateSource` indicates that a source can be generated.

Only the STL and SCL programming languages are supported for blocks. Exceptions are thrown in the following cases:

- Programming language is not STL or SCL
- A file of the same name already exists at the target location

Only the "*.udt" file extension is supported for user data types. Exceptions are thrown in the following cases:

- The file extension is not "*.db" for DBs
- The file extension is not "*.awl" for STL blocks
- The file extension is not "*.scl" for SCL blocks

Program code

Modify the following program code to generate source files from blocks and types:

```
PlcExternalSourceSystemGroup.GenerateSource(IEnumerable<IGenerateSource> plcBlocks,
FileInfo sourceFile, GenerateOptions generateOptions);
// ...
//examples
// ...
var blocks = new List<PlcBlock>(){block1};
var fileInfo = new FileInfo(@"C:\temp\SomePathHere.scl");
PlcExternalSourceSystemGroup systemGroup = ...;
systemGroup.GenerateSource(blocks, fileInfo, GenerateOptions.WithDependencies);
// exports all blocks and with all their dependencies(e.g. called blocks, used DBs or UDTs)
// as ASCII text into the provided source file.
// ...
// or
// ...
var types = new List<PlcType>(){udt1};
var fileInfo = new FileInfo(@"C:\temp\SomePathHere.udt");
PlcExternalSourceSystemGroup systemGroup = ...;
systemGroup.GenerateSource(types, fileInfo, GenerateOptions.WithDependencies );
// exports all data types and their used data types into the provided source file.
// ...
```

See also

Importing configuration data (Page 1212)

5.11.3.19 Generating blocks from source

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Application

You can generate blocks from all external files in the "External source files" group. Only external files with the format ASCII are supported.

Note

Existing blocks are overwritten.

An Exception is thrown if an error occurs during the calling. Before TIA Portal Openness V18, The first 256 characters of each error message are contained in the notification of the Exception. With TIA Portal Openness V18, the first 4096 characters of the message are returned.

The project is reset to the processing state prior to the execution of the `GenerateBlocksFromSource` method.

Program code

Modify the following program code to generate blocks from all external files in the "External source files" group.

```
// Creates a block from an external source file
PlcSoftware plcSoftware = ...;
foreach (PlcExternalSource plcExternalSource in
plcSoftware.ExternalSourceGroup.ExternalSources)
{
plcExternalSource.GenerateBlocksFromSource();
}
```

5.11.3.20 Generating from source of known source format

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online

Application

The TIA Portal Openness API interface supports the generation of block and UDT object. There are conditions where the generated block and UDT are failed to compile.

Typical missing or invalid parts which makes the block and UDT not compilable:

- Reference to a not existing type
- Missing symbol used

The GenerateBlocksFromSource() fails to generate UDT and block with following result:

- A recoverable exception is thrown
- The error message(s) of the generating to the exception's message
- Deletes the generated block/UDT

A new overload of GenerateBlocksFromSource() invoke with GenerateBlockOptions.KeepOnError will lead to the below result:

- Recoverable exception will not be thrown regardless the generating results
- No information about the generation result of any generated block(s)
- Returns with list of IEngineeringObjects of the the generated blocks (with or without generation errors)

If the GenerateBlocksFromSource() invoke with GenerateBlockOptions.None and then generation of block and UDT is error free with following result:

- Recoverable exception will be thrown for failed generation
- Includes the error message(s) of the generating to the exception's message
- Deletes the generated block/UDT

In case of flawless generation, it

- doesn't throw any exception
- doesn't provide information about the generation result of any generated blocks
- returns with an IList of IEngineeringObjects of the the generated blocks
  - this is the same behavior as having called it with GenerateBlockOptions.KeepOnError.

Program code

```
try
{
  IList<IEngineeringObject> generatedObjects =
externalSource.GenerateBlocksFromSource(GenerateBlockOptions.KeepOnError);
  foreach (IEngineeringObject engineeringObject in generatedObjects)
{
  CompilerResult compilerResult = null;
  string objectName = null;
  if (engineeringObject is PlcBlock)
  {
   // handle case for PLcBlock
   // e.g. retrieve the compiler result
   PlcBlock block = (PlcBlock)engineeringObject;
   objectName = block.Name;
   compilerResult = block.Compile();
  }
  else if (engineeringObject is PlcType)
  {
   // handle case for PlcType
   // e.g. retrieve the compiler result
   PlcType plcType = (PlcType)engineeringObject;
   objectName = plcType.Name;
   compilerResult = plcType.Compile();
  }
  // handle the compiler result
  if (compilerResult != null)
  {
   if (compilerResult.State == CompilerResultState.Error)
   {
    Console.WriteLine("Object '{0}' could not be compiled successfully!", objectName);
    Console.WriteLine("Number of compiler errors: {0}", compilerResult.ErrorCount);
    foreach (CompilerResultMessage compilerResultMessage in compilerResult.Messages)
    {
     Console.WriteLine(compilerResultMessage.Description);
    }
   }
   else
   {
    Console.WriteLine("Object '{0}' could be compiled successfully.", objectName);
   }
  }
 }
}
catch (RecoverableException exception)
{
  // handle recoverable exception
  Console.WriteLine(exception.Message);
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.3.21 Deleting user data type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Program code

Modify the following program code to delete a user type:

```
private static void DeleteUserDataType(PlcSoftware plcSoftware)
{
    PlcTypeSystemGroup typeGroup = plcSoftware.TypeGroup;
    PlcTypeComposition dataTypes = typeGroup.Types;
    PlcType dataType = dataTypes.Find("DataTypeName");
    if (dataType != null)
    {
        dataType.Delete();
    }
}
```

See also

Importing configuration data (Page 1212)

5.11.3.22 Accessing PLC system data types

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Introduction

You can use the properties Siemens.Engineering.SW.PlcSystemTypeGroup and Siemens.Engineering.SW.PlcSystemTypeGroupComposition to access a list of system-generated types.

The PlcSystemTypeGroupComposition provides access to the PLC system type groups, while the PlcSystemTypeGroup allows you to access each system's data types.

Program code

Modify the following program code to access the system data types:

```
private static void accessPlcSystemTypeGroup(PlcSystemTypeGroupComposition, m_target)
{
// SystemTypeGroups is a composition of system data types
PlcSystemTypeGroup plcSystemTypeGroup =
m_target.TypeGroup.SystemTypeGroups.FirstOrDefault();
}
```

See also

Opening a project (Page 128)

5.11.3.23 Deleting an external file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  see Connecting to the TIA Portal (Page 82)
- You have opened a project via a TIA Portal Openness application:
  see Opening a project (Page 128)
- PLC is not online

Program code

Modify the following program code to delete an external file in the "External source files" group.

Note

Access to groups in the "External source files" group is not supported.

```
// Deletes an external source file
private static void DeleteExternalSource(PlcSoftware plcSoftware)
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.Find("myExternalsource");
    externalSource.Delete();
}
```

5.11.3.24 Starting the block editor

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- Instance of the TIA Portal is opened with user interface.

Program code

Modify the following program code to start the associated editor for an object reference of the type `PlcBlock` in the TIA Portal instance:

```
//Opens a block in a block editor
private static void StartBlockEditor(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.ShowInEditor();
}
```

Modify the following program code to open the associated editor for an object reference of the type `PlcType` in the TIA Portal instance:

```
//Opens a udt in udt editor
private static void StartPlcTypEditor(PlcSoftware plcSoftware)
{
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find("my_udt");
    udt.ShowInEditor();
}
```

See also

Importing configuration data (Page 1212)

5.11.3.25 Changing blocks using fingerprints

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

The TIA Portal Openness supports fingerprint with name Program Code which compares pure code and ignores inline comments, multi line comments, line breaks and Tabs. The Openness fingerprintId ProgramCode is used to extract the checksum value of the fingerprint "Program Code"

You can use the TIA Portal Openness to detect changes inside of blocks and UDT. You can achieve the functionality by comparing the fingerprints of the object. A fingerprint instance contains a FingerprintId which defines the fingerprint type and the fingerprint value as a string.

All provided fingerprints consider only user input, no compilation result, or any other change made by the system. You can use the FingerprintProvider service to retrieve the instance of fingerprints. The FingerprintProvider service is available for blocks and UDTs, but not for tag tables. The GetFingerprints() is invoked on the fingerprints instance to calculate and return all available fingerprints object. The block or UDT needs to be consistent before calling fingerprints to get the valid fingerprints. Otherwise, a RecoverableException is thrown. When a fingerprint is still invalid after its calculation, a RecoverableException is thrown.

The enumeration FingerprintId lists all kind of fingerprints supported in Openness:

Value	Description
Code	Considers all changes in the code inside the body of the block. It does not consider the compilation result.
Interface	Considers all changes in the interface of a block. Including start values of a DB
Properties	Considers changes in the properties of a block. e.g. name, number
Comments	Considers changes in the comments of a block. In case of OBs the fingerprint also changes when the list of available languages in project language setting changes
LibraryType	Exists when a block is connected to a library type
Texts	With V15 SP1 this fingerprint only exists for Graph blocks
Alarms	Exists when a block uses alarming.
Supervision	Exists when a block contains supervision
TechnologyObject	Exists only for technology object DBs
Events	Exists only for OB

Value	Description
TextualInterface	Exists when the block has a textual interface
ProgramCode	Considers pure code changes and ignores inline comments, multi-line comments, line breaks and tabs

Program code

Modify the following program code to extract the checksum value of the fingerprint “Program Code":

```
var blocks = plcSoftware.BlockGroup.Blocks;
var block1 = blocks.Find("FBDBlock") as CodeBlock;
FingerprintProvider provider = block1.GetService<FingerprintProvider>();
IList<Fingerprint> fingerprints = provider.GetFingerprints();
foreach(var fingerprint in fingerprints)
{
 string fpValue = fingerprint.Value;
 FingerprintId fpId = fingerprint.Id;
 Console.WriteLine("Fingerprint Id:"+ fingerprint.Id+" checksum :"+ fingerprint.Value);
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.3.26 Generating/deleting blocks for user defined pages

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal Openness to create user defined web pages within the web server of PLCs. The data for these user defined pages are stored in specific system generated data blocks of the PLC. You can also edit and delete these blocks manually. But if you modify any data or properties in these data blocks the web server will not function correctly.

In TIA Portal Openness, the new service WebserverUserDefinedPages is added at the DeviceItem PLC. You can use this service to generate the DBs for user defined pages: List<PLCBlock> GenerateBlocks(Arguments). When this function is called the corresponding DBs are generated and a list of generated DBs is returned to user.

There are two possible overloads for this function:

- IList<PlcBlock> WebserverUserDefinedPages.GenerateBlocks(WebDBGenerateOptions GenerateOptions);
- IList<PlcBlock> WebserverUserDefinedPages.GenerateBlocks(System.IO.DirectoryInfo htmlDirectory, System.IO.FileInfo defaultHTMLPage, string applicationName, WebDBGenerateOptionsGenerateOptions);

The parameter values for HTML directory, HTML page and application name are not stored in the project data. So the values of the corresponding dynamic attributes are not changed.

Parameter	Data type	Type	Descriptions
GenerateOptions	WebDBGenerateOptions (enumeration)	Mandatory	Possible values: • None - If DBs for user defined pages already have been created before, no action is performed and an exception is thrown • Override - If DBs for user defined pages already have been created before, they are deleted and the new DBs are generated
htmlDirectory	System.IO.DirectoryInfo	Optional	Specifies the HTML directory independently from the value of the dynamic attribute WebserverHTMLDirectory
defaultHTMLPage	System.IO.FileInfo	Optional	Specifies the default HTML page independently from the value of the dynamic attribute WebserverDefaultHTMLPage
applicationName	string	Optional	Specifies the application name independently from the value of the dynamic attribute WebserverApplicationName

The GenerateBlocks() will throw recoverable exception:

- If web server is disabled (WebserverActive == False) - "Webserver has to be enabled"
- If WebserverHTMLDirectory is empty or the path is invalid - "HTML directory has invalid path"
- If WebserverdefaultHTMLPage is empty or the path is invalid - "Default HTML page has invalid path"
- When the WebserverHTMLDirectory is too long - "HTML directory has too large data"
- The Application Name is invalid - "Application Name is invalid"
- The Dynamic Content is invalid - "Dynamic Content is invalid"
- The Control DB number is invalid - "Control DB number is invalid"

- The DB Start Number is invalid - "DB Start Number is invalid"
- Blocks are already present: If the user tries to generate a block with a name which already exists- ""Delete existing blocks before generating new blocks" (Only thrown if WebDBGenerateOptions.None is used)

There is no specific function to delete all blocks connected to the user defined pages. You have to manually delete the blocks that are handed back to them by the generate action or navigate to the corresponding DBs as it is already possible in TIA Portal Openness.

Program code

Modify the following program code to generate a block for user defined page:

```
DirectoryInfo htmlDirectory = ...;
FileInfo defaultHTMLPage = ...;
string applicationName = ...;
DeviceItem deviceItem = ...;
var WebserverUserDefinedPagesService = deviceItem.GetService<WebserverUserDefinedPages>();
IList<PlcBlock> block =
WebserverUserDefinedPagesService.GenerateBlocks(WebDBGenerateOptions.None);
IList<PlcBlock> blocks = WebserverUserDefinedPagesService.GenerateBlocks(htmlDirectory,
defaultHTMLPage, applicationName, WebDBGenerateOptions.Override);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.3.27 Writing access for OB block priority attribute

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to provide write access for block property especially for OBs. You can change the priority of a block both in unit and non-unt programming model using TIA Portal Openness.

The following property is available to access for block:

Property name	Data type	Access
PriorityNumber	Integer	Read/Write

Program code

Modify the following program code to set the priority for OB block:

```
PlcSoftware plcSoftware = ...;
PlcBlock obBlock = plcSoftware.BlockGroup.Blocks.Find("CyclicInterrupt");
obBlock.SetAttribute("PriorityNumber", 14);
```

See also

Opening a project (Page 128)

5.11.3.28 Generating block/UDT from external source file in specific user group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project via a TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to generate blocks/ udts under user specific groups in both PLC programming model and unit programming model..

Functions

Method name with signature

- Public IList<IEngineeringObject> PlcExternalSource.GenerateBlocksFromSource(PlcBlockUserGroup blockuserGroup, GenerateBlockOption generateBlockOption)
- Public IList<IEngineeringObject> PlcExternalSource.GenerateBlocksFromSource(PlcBlockUserGroup blockuserGroup, GenerateBlockOption generateBlockOption)

Program code: Generation of blocks from source under specific user groups

Modify the following program code to retrieve the user group information under which the blocks to be generated:

```
Siemens.Engineering.SW.Blocks.PlcBlockUserGroup folder =
plc.BlockGroup.Groups.Find("Group_1");
```

Modify the following program code to generate blocks under groups without any options.

```
plc.ExternalSourceGroup.ExternalSources.Find("Block_1.scl").GenerateBlocksFromSource(folde
r, GenerateBlockOption.None);
```

Modify the following program code to generate blocks with Options as KeepOnError:

```
plc.ExternalSourceGroup.ExternalSources.Find("Block_1.scl")
GenerateBlocksFromSource(folder, GenerateBlockOption.KeepOnError);
```

Program code: Generation of UDTs from source under specific user groups

```
Siemens.Engineering.SW.Types.PlcTypeUserGroup folder1 =
plc.TypeGroup.Groups.Find("Group_1");
```

Modify the following program code to generate UDTs under groups without any Options:

```
plc.ExternalSourceGroup.ExternalSources.Find("User_data_type_1.udt").GenerateBlocksFromSou
rce(folder1, GenerateBlockOption.None);
```

Modify the following program code to generate blocks with Options as KeepOnError:

```
plc.ExternalSourceGroup.ExternalSources.Find("User_data_type_1.udt").GenerateBlocksFromSou
rce(folder1,GenerateBlockOption. KeepOnError);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.3.29 Generating loadable file for blocks outside of software units

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- The TIA Portal project is open
  See Opening a Project (Page 128)
- The PLC must be compile clean

Introduction

You can use the TIA Portal Openness to generate a loadable file for blocks outside of software units via the following method `GenerateLoadable(FileInfo path, IEnumerable<PlcBlock> blocks, TargetOption targetOption)`

Note

The generation of a loadable file for blocks is supported for PLC1518 or its subtype whose firmware is V2.8 or higher.

Restriction

The following restrictions apply when selecting the block:

- The block must not be a system block
- The block must not be a PRODIAG block
- The block must not be a failsafe block
- If the block is an OB, it must not be of any other type than ProgramCycle
- The block must not have a dynamic binding
- The block must not be a technology object (TO)

Parameter and Enum

The GenerateLoadable() accepts the following parameters to generate a loadable file for blocks:

Parameter name	Data type	Description	Note
path	FileInfo	Specifies the name and the storage location for the loadable file	Already existing loadable files cannot be replaced and the containing directory cannot be created.
blocks	System.Collections.Generic.IEnumerable<Siemens.Engineering.SW.Blocks.PlcBlock>	Specifies the list of program blocks outside of software units to generate the loadable file from	
targetOption	Siemens.Engineering.SW.Loader.TargetOption	Specifies whether the loadable file is to be created for a real PLC or for a simulated PLC	If TargetOption.PlcSim is passed, all blocks must have been compiled with simulation support

The TargetOption parameter uses the following enum values to generate a loadable file:

Value	Description
TargetOption.None	The default value, should not be used.
TargetOption.Plc	To generate a loadable file for a real PLC.
TargetOption.PlcSim	To generate a loadable file for a simulated PLC.

Program code

To generate a loadable file for blocks outside of software units modify the following program code:

```
private void GenerateLoadableProgramBlock(string loadablePath)
{
// Getting an access to the PLC device of the project
PlcSoftware plc = GetPlc();
// The GetService() will return null if the PLC is not supported.
var loadableProviderService = plc.GetService<LoadableProvider>();
// Create a variable file with name "file" to access the loadable path
var file  = new FileInfo(loadablePath);
// Create a variable with name "blocks" for a list of program blocks
// from the non-software unit program.
var blocks = new List<PlcBlock>
{
// Accessing a specific block
// with name "Block_1" under plc
plc.BlockGroup.Blocks.Find("Block_1") ?? throw new InvalidOperationException()
};
// The loadableProviderService generates a loadable file for program block
// by calling the GenerateLoadable() with respective parameters.
try
{
loadableProviderService.GenerateLoadable(file, blocks, TargetOption.Plc);
}
catch (EngineeringTargetInvocationException e)
{
// An EngineeringTargetInvocation exception is thrown
// if the generation of the loadable file fails.
// The exception message contains hints on
// what caused the loadable file creation to fail.
Console.WriteLine("Generating the loadable file failed.");
Console.WriteLine(e);
}
}
```

Exception

The following possible exception message are listed below:

- It is only possible to create a loadable file from a whole software unit, not from a single block inside of a unit.
- The selected blocks are from different PLCs.
- Destination directory not found: '{0}'.
- It is not possible to create a loadable file from failsafe block: '{0}'.
- Destination file already exists: '{0}'.
- The loadable file is not accessible.
- An internal failure occured.
- The PLC has to be compiled.

- Internal serialization of loadable file failed.
- Internal serialization failed, because loadable file was disposed.
- The specified path, file name, or both exceed the system-defined maximum length.
- The PLC '{0}' is not supported.
- The parent PLC of the service and of the target are different.
- Some block(s) cannot be simulated. If a block is a library block, use a library with simulation support. Otherwise, select the option "Support simulation during block compilation" in the project properties and recompile the block.
- It is not possible to create a loadable file from a system block: '{0}'.
- Access to file is denied.
- Software units containing PLC data types are not supported.
- It is not possible to create a loadable file from selected object(s).
- It is not possible to create a loadable file from OB '{0}', because it does not have the event class 'Program cycle.
- It is not possible to create a loadable file from a Prodiag block: '{0}'.
- It is not possible to create a loadable file from selected object: '{0}.
- It is not possible to create a loadable file from block '{0}' with dynamic binding enabled.

See also

Generating loadable file for software units (Page 624)

5.11.4 Technology objects

5.11.4.1 Overview of functions for technology objects

TIA Portal Openness supports a selection of technology object functions for defined tasks that you can call outside the TIA Portal by means of the Public API.

In the following chapters you will find code sample, that can be adapted to your openness program.

Functions

The following functions are available for technology objects:

- Querying the composition of technology objects (Page 530)
- Creating technology object (Page 530)
- Deleting technology object (Page 531)
- Creating group for technology objects (Page 532)

- Deleting group for technology objects (Page 533)
- Compiling technology object (Page 533)
- Enumerating technology object (Page 535)
- Enumerating groups for technology objects (Page 535)
- Finding technology object (Page 537)
- Enumerating parameters of technology object (Page 537)
- Finding parameters of technology object (Page 538)
- Reading parameters of technology object (Page 539)
- Writing parameters of technology object (Page 540)
- Exporting technology objects (Page 1406)
- Importing technology objects (Page 1408)
- Connecting to Hardware components (Page 549)

See also

Standard libraries (Page 74)

Applications (Page 50)

TIA Portal Openness object model (Page 53)

5.11.4.2 Overview of technology objects and versions

Technology objects

The following table shows the available technology objects in the Public API.

CPU	Technology	Technology object	Version of technology object	CPU FW
S7-1200	Motion Control	TO_PositioningAxis	≥ V6.0	≥ V4.2
		TO_CommandTable		
	PID Control	PID_Compact	≥ V2.3	≥ V4.2
		PID_3Step	V2.3	
		PID_Temp	V1.1	

CPU	Technology	Technology object	Version of technology object	CPU FW
S7-1500	Motion Control	TO_SpeedAxis	≥ V3.0	≥ V2.0
		TO_PositioningAxis		
		TO_ExternalEncoder		
		TO_SynchronousAxis		
		TO_OutputCam		
		TO_CamTrack		
		TO_MeasuringInput		
		TO_Cam (S7-1500T)[1]		
		TO_Kinematics (S7-1500T)	≥ V4.0	≥ V2.6
		TO_LeadingAxisProxy (S7-1500T)	≥ V5.0	≥ V2.8
		TO_Cam_10k (S7-1500T)[1]	≥ V6.0	≥ V2.9
		TO_Interpreter (S7-1500T)	≥ V8.0	≥ V3.1
		TO_InterpreterMapping (S7-1500T)[2]		
		TO_InterpreterProgram (S7-1500T)[2]		
	Counting and measurement	High_Speed_Counter	≥ V4.1	Any
		SSI_Absolute_Encoder	≥ V3.1	
	PID Control	PID_Compact	≥ V2.3	≥ V2.0
		PID_3Step	V2.3	
		PID_Temp	V1.1	
		CONT_C	V1.1	Any
		CONT_S		
		TCONT_CP		
		TCONT_S		
S7-300/400	PID Control	CONT_C	V1.1	Any
		CONT_S		
		TCONT_CP		
		TCONT_S		
		TUN_EC[2]		
		TUN_ES[2]		
		PID_CP[2]	V2.0	
		PID_ES[2]		
	EMC	AXIS_REF	V2.0	

[1] The technology object does not support the following Openness functions: Writing parameters.

[2] The technology object does not support the following Openness functions: Enumerating parameters, Finding parameters, Reading parameters, Writing parameters.

See also

S7-1500 Motion Control (Page 549)

5.11.4.3 Overview of data types

The data types of technology object parameters in TIA Portal are mapped to C# data types in the Public API.

Data types

The following table shows the data type mapping:

Format	Data type in TIA Portal	Data type in C#
Binary numbers	Bool	bool
	BBool	bool
	Byte	byte
	Word	ushort
	DWord	uint
	LWord	ulong
Integers	SInt	sbyte
	Int	short
	Dint	int
	LInt	long
	USInt	byte
	UInt	ushort
	UDint	uint
	ULInt	ulong
Floating-point numbers	Real	float
	LReal	double
	Time	double
Character strings	Char	char
	WChar	char
	String	string
	WString	string
Hardware data types	HW_*	ushort
	Block_*	ushort

* Placeholder for device type extension in TIA Portal project

5.11.4.4 Querying the composition of technology objects

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)

Program code

Modify the following program code to get all technology objects of a PLC:

```
// Retrieves all technology objects of a PLC
private static void GetTechnologicalObjectsOfPLC(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBGroup technologicalObjectGroup =
plcSoftware.TechnologicalObjectGroup;
    TechnologicalInstanceDBComposition technologicalObjects =
technologicalObjectGroup.TechnologicalObjects;
}
```

See also

Standard libraries (Page 74)

5.11.4.5 Creating technology object

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)

Application

Only technology objects that are listed in the section Overview of technology objects and versions (Page 527) can be created. An exception is thrown for unsupported technology objects or invalid parameters. See also Handling exceptions (Page 1187).

Program code

Modify the following program code to create a technology object and add it to an existing PLC:

```
// Create a technology object and add to technology object composition
private static void CreateTechnologicalObject(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;

    string nameOfTO = "PID_Compact_1"; // How the technology object should be named
    string typeOfTO = "PID_Compact"; // How the technology object type is called, e.g. in
    // "Add new technology object"-dialog
    Version versionOfTO = new Version("2.3"); // Version of technology object
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Create(nameOfTO,
typeOfTO, versionOfTO);
}
```

Possible values and combinations of name, type and version of the technology object can be found in the section Overview of technology objects and versions (Page 527).

See also

Standard libraries (Page 74)

S7-1500 Motion Control (Page 549)

5.11.4.6 Deleting technology object

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)
- The technology object exists.
  See Finding technology object (Page 537)

Program code

Modify the following program code to delete a technology object:

```
// Delete a technology object from DB composition and from PLC
private static void DeleteTechnologicalObject(TechnologicalInstanceDB technologicalObject)
{
    technologicalObject.Delete();
}
```

See also

Standard libraries (Page 74)

5.11.4.7 Creating group for technology objects

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)

Program code

Modify the following program code to create a group for technology objects:

```
private static void CreateTechnologicalObjectGroup(PlcSoftware plcsoftware)
//Create a user group for technology objects
{
    TechnologicalInstanceDBGroup systemGroup = plcsoftware.TechnologicalObjectGroup;
    TechnologicalInstanceDBUserGroupComposition groups = systemGroup.Groups;
    TechnologicalInstanceDBUserGroup myGroup = groups.Create("MySubGroupName");
}
```

5.11.4.8 Deleting group for technology objects

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)

Program code

Modify the following program code to delete a group for technology objects:

```
private static void DeleteTechnologicalObjectGroup(PlcSoftware plcsoftware)
// Delete groups for technology objects
{
    TechnologicalInstanceDBGroup group
        = plcsoftware.TechnologicalObjectGroup.Groups.Find("myGroup");
    TechnologicalInstanceDBUserGroupComposition subgroups = group.Groups;
    TechnologicalInstanceDBUserGroupComposition subgroup = subgroups.Find("myUserGroup");
    if (subgroup != null)
    {
        subgroup.Delete();
    }
}
```

5.11.4.9 Compiling technology object

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)
- The technology object exists.
  See Creating technology object (Page 530)

Program code: Compiling a technology object

Modify the following program code to compile a technology object:

```
// Compile a single technology object
private static void CompileSingleTechnologicalObject(TechnologicalInstanceDB
technologicalObject)
{
    ICompilable singleCompile = technologicalObject.GetService<ICompilable>();
    CompilerResult compileResult = singleCompile.Compile();
}
```

Program code: Compiling the technology object group

Modify the following program code to compile the technology object group:

```
// Compile technology object group
private static void CompileTechnologicalObjectGroup(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBGroup technologicalObjectGroup =
plcSoftware.TechnologicalObjectGroup;
    ICompilable groupCompile = technologicalObjectGroup.GetService<ICompilable>();
    CompilerResult compileResult = groupCompile.Compile();
}
```

Compile results

Technology objects compilation results are stored recursively.

You can find an example of recursive evaluation of compilation results in the section "Compiling a project (Page 153)".

Note

Further parameters

Imported data will be expanded after compiling. For example, You can use Openness to create motion control technology objects with different versions. After the compiling of technology objects, the motion control versions of other technology objects are adapted to those of the last technology object created.

See also

Standard libraries (Page 74)

5.11.4.10 Enumerating technology object

Requirement

You can enumerate the top level TOs like "TO_Axis" and "TO_Cam". To find sub level TOs like "TO_OutputCam" see Creating and finding TO_OutputCam, TO_CamTrack and TO_MeasuringInput (Page 549).

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)

Program code

Modify the following program code to enumerate technology objects:

```
// Enumerate all technology objects
private static void EnumerateTechnologicalObjects(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;
    foreach (TechnologicalInstanceDB technologicalObject in technologicalObjects)
    {
        // Do something ...
    }
}
```

See also

Standard libraries (Page 74)

S7-1500 Motion Control (Page 549)

5.11.4.11 Enumerating groups for technology objects

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)

Application

Subgroups are taken into account recursively for enumeration.

Program code: Enumerating all groups

Modify the following program code to enumerate the groups for technology objects:

```
//Enumerates all groups for technology objects
private static void EnumerateRecursively(PlcSoftware sw)
{
    foreach (TechnologicalInstanceDBUserGroup group in sw.TechnologicalObjectGroup.Groups)
    {
         EnumerateRecursivelySubGroups (group);
    }
}

//Enumerates all included sub groups
private static void EnumerateRecursivelySubGroups(TechnologicalInstanceDBUserGroup group)
{
    foreach (TechnologicalInstanceDBUserGroup subGroup in group.Groups)
    {
         EnumerateRecursivelySubGroups (subGroup);
    // recursion
    }
}
```

Program code: Accessing a group

Modify the following program code to access a selected user-defined group for technology objects:

```
//Gives individual access to a specific  group for technology objects
private static void AccessTechnologicalInstanceDBUserGroup(PlcSoftware sw)
{
    TechnologicalInstanceDBUserGroupComposition groups =
sw.TechnologicalObjectGroup.Groups;
    TechnologicalInstanceDBUserGroup userGroup = groups.Find("MyUserfolder");
}
```

5.11.4.12 Finding technology object

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)

Program code

Modify the following program code to find a specific technology object:

```
// Find a specific technology object by its name
private static void FindTechnologicalObject(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Find(nameOfTO);
}
```

See also

Standard libraries (Page 74)

5.11.4.13 Enumerating parameters of technology object

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)
- A technology object exists.
  See Creating technology object (Page 530) or Finding parameters of technology object (Page 538)
- The technology object (Page 527) supports this function.

Program code

Modify the following program code to enumerate parameters of a specific technology object:

```
// Enumerate parameters of a technology object
private static void EnumerateParameters(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    foreach (TechnologicalParameter parameter in technologicalObject.Parameters)
    {
        // Do something ...
    }
}
```

See also

Standard libraries (Page 74)

Finding technology object (Page 537)

5.11.4.14 Finding parameters of technology object

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)
- A technology object exists.
  See Creating technology object (Page 530)
- The technology object (Page 527) supports this function.

Program code

Modify the following program code to find parameters of a specific technology object:

```
// Find parameters of a technology object
private static void FindParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);
}
```

Parameters of different technology objects

Parameters of S7-1200 Motion Control (Page 541)

Parameters of S7-1500 Motion Control (Page 549)

Parameters of PID Control (Page 578)

Parameters of Counting (Page 578)

Parameters of Easy Motion Control (Page 578)

See also

Standard libraries (Page 74)

Finding technology object (Page 537)

5.11.4.15 Reading parameters of technology object

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)
- A technology object exists.
  See Creating technology object (Page 530)
- The technology object (Page 527) supports this function.

Program code

Modify the following program code to read parameters of a specific technology object:

```
// Read parameters of a technology object
private static void ReadParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);

    // Read from parameter
    string name = parameter.Name;
    object value = parameter.Value;
}
```

See also

Standard libraries (Page 74)

Finding technology object (Page 537)

5.11.4.16 Writing parameters of technology object

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)
- A technology object exists.
  See Creating technology object (Page 530)
- The technology object (Page 527) supports this function.

Exception

An EngineeringException is thrown if:

- You set a new value for a parameter that does not provide write access.
- A new value for a parameter is of an unsupported type.

Program code

Modify the following program code to write parameters of a specific technology object:

```
// Write parameters of a technology object
private static void WriteParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);

    // Write to parameter if the value is writable
    object value = 3.0;
    parameter.Value = value;
}
```

Parameters of different technology objects

Parameters of S7-1200 Motion Control (Page 541)

Parameters of S7-1500 Motion Control (Page 549)

Parameters of PID Control (Page 578)

Parameters of Counting (Page 578)

Parameters of Easy Motion Control (Page 578)

See also

Standard libraries (Page 74)

Finding technology object (Page 537)

5.11.4.17 S7-1200 Motion Control

Connecting PTO-Outputs

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).

- A S7-1200 PLC with PTO-Outputs is determined in the project.
- The technology object exists.
  See Creating technology object (Page 530).

Program code

Modify the following program code to connect a PTO-Output to the "TO_PositioningAxis".

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingDrive(TechnologicalInstanceDB technologicalObject)
{
   //Set axis to PTO mode
   technologicalObject.Parameters.Find("Actor.Type").Value = 2;
   //Set PTO-Output
   technologicalObject.Parameters.Find("_Actor.Interface.PTO").Value = 0; // select
Pulse_1
   // 0 = Pulse_1
   // 1 = Pulse_2
   // 2 = Pulse_3
   // 3 = Pulse_4
}
```

Connecting PROFIdrives by hardware address

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1200 PLC is determined in the project.
- A PROFIdrive is available in the project and connected with the S7-1200 PLC.
- The technology object exists.
  See Creating technology object (Page 530).

Program code

Modify the following program code to connect a PROFIdrive by hardware address to the "TO_PositioningAxis".

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingDrive(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set axis to drive mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 0;

    //Set connection to address of drive. The output will be set automatically.
    technologicalObject.Parameters.Find("_Actor.Interface.ProfiDriveIn").Value = "%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Connecting encoders for PROFIdrives by hardware address

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1200 PLC is determined in the project.
- A PROFIdrive is available in the project and connected with the S7-1200 PLC.
- The technology object exists.
  See Creating technology object (Page 530).

Program code

Modify the following program code to connect an encoder by hardware address to the "TO_PositioningAxis":

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set the encoder mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;

    //Set axis to use PROFINET encoder
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value = 0;

    //Set connection to address of drive. The output will be set automatically.
    technologicalObject.Parameters.Find("_Sensor[1].Interface.ProfiDriveIn").Value =
"%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Connecting analog drives by hardware address

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1200 PLC is determined in the project.
- An analog drive is available in the project and connected with the S7-1200 PLC.
- The technology object exists.
  See Creating technology object (Page 530).

Program code

Modify the following program code to connect an analog drive by hardware address to the "TO_PositioningAxis":

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set axis to drive mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 0;

    //Set connection to analog address of drive
    technologicalObject.Parameters.Find("_Actor.Interface.Analog").Value = "%QW64";
}
```

Connecting encoders for analog drives by hardware address

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1200 PLC is determined in the project.
- An analog drive is available in the project and connected with the S7-1200 PLC.
- The technology object exists.
  See Creating technology object (Page 530).

Program code

Modify the following program code to connect an encoder by hardware address to the "TO_PositioningAxis":

```
//An instance of the technology object axis is already available in the program before
//Connecting by High Speed Counter mode
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set encoder for high-speed counter mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
4;
    technologicalObject.Parameters.Find("_Sensor[1].Interface.HSC.Name").Value = "HSC_1";
}

    //An instance of the technology object axis is already available in the program before
    //Connecting by PROFINET/PROFIBUS telegram
    private static void ConnectingEncoder(TechnologicalInstanceDB
    technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;
    //Set encoder for PROFINET/PROFIBUS mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value =
"Encoder";
    technologicalObject.Parameters.Find("_Sensor[1].Interface.ProfiDriveIn").Value =
"%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Connecting drives by data block

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1200 PLC is determined in the project.

- A data block is available in the project and set to "Not optimized".
  For the PROFIdrive axis type, the data block contains a tag of the type e. g. PD_TEL3.
  For an analog drive, the data block contains a tag with the word data type.
- The technology object exists.
  See Creating technology object (Page 530).

Program code

Modify the following program code to connect a PROFIdrive by data block to the "TO_PositioningAxis".

```
//An instance of the technology object axis is already available in the program before
private static void ConfigureDrivewithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set axis to data block mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 1;

    //Set the tag in the data block
    technologicalObject.Parameters.Find("_Actor.Interface.DataBlock").Value =
"Data_block_1.Member_of_type_PD_TEL3";
}
```

Program code

Modify the following program code to connect an analog drive by data block to the "TO_PositioningAxis".

```
//An instance of the technology object axis is already available in the program before
//Connecting an analog drive with data block.
private static void ConfigureDrivewithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to analog mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set the tag in the data block
    technologicalObject.Parameters.Find("_Actor.Interface.Analog").Value =
"Data_block_1.Static_1";
}
```

Connecting encoders by data block

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1200 PLC is determined in the project.
- A data block is available in the project and set to "Not optimized".
  In case of PROFIdrive the data block contains a tag of the type e. g. PD_TEL3
- The technology object exists.
  See Creating technology object (Page 530).

Program code

Modify the following program code to connect an encoder by data block:

```
//An instance of the technology object axis is already available in the program before
private static void ConfigureEncoderwithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to PROFIdrive mode depending by axis type. 1 = PROFIdrive, 0 = Analog Drive.
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set the encoder mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;

    //Set axis to data block mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value = 1;

    //Set the tag in the data block. For PD_TEL3 and PD_TEL4 "Encoder1" or "Encoder2".
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataBlock").Value =
"Data_block_1.Member_of_Type_PD_TEL3";
}
```

Parameters for TO_PositioningAxis and TO_CommandTable

You can find a list of all available variables in SIMATIC STEP 7 S7-1200 Motion Control function manual on the internet (https://support.industry.siemens.com/cs/ww/en/view/109773400).

Note

In TIA Portal in the Parameter view of the technology object configuration you can find the column "Name in Openness".

5.11.4.18 S7-1500 Motion Control

Creating and finding TO_OutputCam, TO_CamTrack and TO_MeasuringInput

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_PositioningAxis, TO_SynchronousAxis or TO_ExternalEncoder is determined in the project.

Application

The output cam, cam track and measuring input technology objects are associated with positioning axis, synchronous axis or external encoder technology objects. In order to access an output cam, cam track or measuring input technology object you use the service OutputCamMeasuringInputContainer.

Program code: Creating and finding output cam, cam track and measuring input technology objects

Modify the following program code to create or find an output cam, cam track or measuring input technology object:

```
/*An instance of the technology object under which the TO_OutputCam, TO_CamTrack or
TO_MeasuringInput should be created is already available in the program before*/
private static void CreateFind_OutputcamCamtrackMeasuringinput(TechnologicalInstanceDB
technologyObject)
{
    //Retrieve service OutputCamMeasuringInputContainer
    OutputCamMeasuringInputContainer container =
    technologyObject.GetService<OutputCamMeasuringInputContainer>();
    //Get access to TO_OutputCam / TO_CamTrack container
    TechnologicalInstanceDBComposition outputcamCamtrackContainer = container.OutputCams;

    //Find technology object TO_OutputCam or TO_CamTrack
    TechnologicalInstanceDB outputCam = outputcamCamtrackContainer.Find("OutputCamName");
    TechnologicalInstanceDB camTrack = outputcamCamtrackContainer.Find("CamTrackName");

    //Create new technology object TO_OutputCam or TO_CamTrack
    TechnologicalInstanceDB newOutputCam =
    outputcamCamtrackContainer.Create("NewOutputCamName", "TO_OutputCam",
    new Version(3, 0));
    TechnologicalInstanceDB newCamTrack =
    outputcamCamtrackContainer.Create("NewCamTrackName", "TO_CamTrack", new Version(3, 0));

    //Get access to TO_MeasuringInput container
    TechnologicalInstanceDBComposition measuringInputContainer = container.MeasuringInputs;

    //Find technology object TO_MeasuringInput
    TechnologicalInstanceDB measuringInput =
    measuringInputContainer.Find("MeasuringInputName");

    //Create new technology object TO_MeasuringInput
    TechnologicalInstanceDB newMeasuringInput =
    measuringInputContainer.Create("NewMeasuringInput", "TO_MeasuringInput",
    new Version(3, 0));
}
```

You create the S7-1500 technology objects OutputCam, CamTrack and MeasuringInput as a master copy from Siemens.Engineering.Library.MasterCopies.MasterCopy.

```
/*An instance of the technology object under which the TO_OutputCam, TO_CamTrack or
TO_MeasuringInput should be created is already available in the program before*/
// sourceMasterCopy contains master copy object output cam
private static void CreateFind_OutputcamCamtrackMeasuringinput(TechnologicalInstanceDB
technologyObject, MasterCopy sourceMasterCopy)
{
    //Retrieve service OutputCamMeasuringInputContainer
    OutputCamMeasuringInputContainer container =
technologyObject.GetService<OutputCamMeasuringInputContainer>();

    TechnologicalInstanceDB result = container.OutputCams.CreateFrom(sourceMasterCopy);

}
```

Parameters of S7-1500 Motion Control

Parameters of S7-1500 Motion Control

Most parameters of S7-1500 Motion Control technology objects are directly mapped to data block tags, but there are also some additional parameters that do not map directly to data blocks.

Parameters mapped directly to technology object data block tags

You have access to all technology object data block tags as described in general except of:

- Read-only tags
- Tags of data type VREF
- Tags of "InternalToTrace" structure
- Tags of "ControlPanel" structure

Some technology parameters that map to read-only data block tags are writeable in the PublicAPI. The allowed values are the same ones as for the underlying data block tags.

The affected parameters are listed for all technology objects in the following chapters.

Program code

Modify the following program code to access the directly mapped parameters:

```
//An instance of the technology object is already available in the program before
private static void ReadWriteDataBlockTag(TechnologicalInstanceDB technologyObject)
{
    //Read value from data block tag "ReferenceSpeed"
    double value =
    (double)technologyObject.Parameters.Find("Actor.DriveParameter.ReferenceSpeed").Value;

    //Write data block tag "ReferenceSpeed"
    technologyObject.Parameters.Find("Actor.DriveParameter.ReferenceSpeed").Value = 3000.0;
}
```

Parameters not mapped directly to technology object data block tags

S7-1500 Motion Control technology objects are connected to other devices or technology objects. The information about this connections is stored in additional parameters as well as informations about units of measure. Additional paremeters are not dircetly mapped to technology object data block tags.

The affected parameters are listed for all technology objects in the following chapters.

Program code: Additional parameters

Modify the following program code to access the additional parameters:

```
//An instance of the technology object is already available in the program before
private static void ReadWriteAdditionalParameter(TechnologicalInstanceDB technologyObject)
{
    //Read additional parameter "_Properties.MotionType"
    uint value = (uint)technologyObject.Parameters.Find("_Properties.MotionType").Value;

    //Write additional parameter "_Properties.MotionType"
    technologyObject.Parameters.Find("_Properties.MotionType").Value = 1;
}
```

Additional information

You can find additional information in SIMATIC S7-1500 Motion Control function manuals:

https://support.industry.siemens.com/cs/ww/en/view/109751049 (https://support.industry.siemens.com/cs/ww/en/view/109751049)

Axis and Encoder

Parameters mapped directly to technology object data block tags

For axis and external encoder technology objects, the following read-only data block tags are writable in the PublicAPI.

Name in Openness	Name in function view	Possible value	Data type in Openness	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_ExternalEncoder
Actor.Type	Drive type	0: Analog output 1: PROFIdrive telegram	int	X	X	-
Actor.Interface.EnableDriveOutput	"Enable output" for analog drives	True, False	bool	X	X	-
Actor.Interface.DriveReadyInput	"Ready input" for analog drives	True, False	bool	X	X	-
Actor.Interface.EnableTorqueData	Torque data	True, False	bool	-	X	-
VirtualAxis.Mode	Virtual axis	0, 1	uint	X	X	-
Sensor[n].Existent[1)]	Use encoder	True, False	bool	-	X	-
Sensor[n].Interface.Number[1)]	Number of the encoder in the telegram	1 ... 2	uint	-	X	-
Sensor[n].Type[1)]	Encoder type	0: Incremental 1: Absolute 2: Cyclic absolute	int	-	X	-
Sensor.Interface.Number	Number of the encoder in the telegram	1 ... 2	uint	-	-	X
Sensor.Type	Encoder type	0: Incremental 1: Absolute 2: Cyclic absolute	int	-	-	X
CrossPlcSynchronousOperation.Interface[n].EnableLeadingValueOutput[5)]	Provide cross-PLC leading value	True, False	bool	-	X	X

[1)] S7-1500 PLC: n=1; S7-1500T PLC: 1≤n≤4

[5)] V5.0 n=1; ≥V6.0 1≤n≤8

Parameters not mapped directly to technology object data block tags

For axis and external encoder technology objects, the following additional parameters are available.

Name in Openness	Name in function view	Possible value	Data type in Openness	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_ExternalEncoder
_Actor.DataAdaptionOffline	Automatically apply...	True, False	bool	X	X	-
_Actor.Interface.Telegram	Drive telegram	Telegram number[2)]	uint	X	X	-
_Actor.Interface.EnableDriveOutputAddress	Drive output address	PublicAPI-object	SW.Tags.PlcTag	X	X	-
_Actor.Interface.DriveReadyInputAddress	Drive ready input address	PublicAPI-object	SW.Tags.PlcTag	X	X	-
_Units.LengthUnit	Position units	See tag Units.LengthUnit[3)]	uint	-	X	X
_Units.VelocityUnit	Velocity units	See tag Units.VelocityUnit[3)]	uint	X	X	X
_Units.TorqueUnit	Torque units	See tag Units.TorqueUnit[3)]	uint	X	X	-
_Units.ForceUnit	Force units	See tag Units.ForceUnit[3)]	uint	-	X	-
_Sensor[n].DataAdaptionOffline[1)]	Automatically apply...	True, False	bool	-	X	-
_Sensor[n].Interface.Telegram[1)]	Encoder telegramm	Telegram number[2)]	uint	-	X	-
_Sensor[n].ActiveHoming.DigitalInputAddress[1)]	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_Sensor[n].PassiveHoming.DigitalInputAddress[1)]	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_Sensor.DataAdaptionOffline	Automatically apply...	True, False	bool	-	-	X
_Sensor.Interface.Telegram	Encoder telegram	Telegram number[2)]	uint	-	-	X
_Sensor.ActiveHoming.DigitalInputAddress	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	-	X
_Sensor.PassiveHoming.DigitalInputAddress	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	-	X

Name in Openness	Name in function view	Possible value	Data type in Openness	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_ExternalEncoder
_Properties.MotionType	Axis type respectively "Technological unit of the position"	0: Linear 1: Rotary	int	-	X	X
_Properties.UseHighResolutionPositionValues[4]	Use position values with higher resolution	True, False	bool	-	X	X
_PositionLimits_HW.MinSwitchAddress	Hardware low limit switch input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_PositionLimits_HW.MaxSwitchAddress	Hardware high limit switch input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_CrossPlcSynchronousOperation.Interface[n]. AddressOut[5]	Output address for the leading value telegram	PublicAPI-object	SW.Tags.PlcTag	-	X	X
_CrossPlcSynchronousOperation. ActivateLocalLeadingValueDelayTimeCalculation[4]	Allow system calculation	True, False	bool	-	X	X

1) S7-1500 PLC: n=1; S7-1500T PLC: 1≤n≤4

2) possible values are described in the function manual S7-1500 Motion Control on chapter PROFIdrive telegrams

3) possible values are described in the function manual S7-1500 Motion Control on chapter units tags (TO)

4) ≥V5.0

5) V5.0 n=1; ≥V6.0 1≤n≤8

Leading axis proxy

Parameters mapped directly to technology object data block tags

For leading axis proxy technology object, no read-only data block tags are writable in the PublicAPI.

Parameters not mapped directly to technology object data block tags

For leading axis proxy technology object, the following additional parameters are available.

Name in Openness	Name in function view	Possible value	Data type
_Interface.AddressIn[1)]	Transfer area	PublicAPI-object	SW.Tags.PlcTag

1) S7-1500T PLC

Measuring input

Parameters mapped directly to technology object data block tags

For measuring input technology object, the following read-only data block tags are writable in the PublicAPI.

Name in Openness	Name in function view	Possible value	Data type in Openness
Parameter.MeasuringInput-Type	Measuring input type	0: Measurement by using time stamp (STANDARD) 1: Measurement by using PROFIdrive telegrams of drive or external encoder (PROFIDRIVE) 2: Measurement by listening to measuring input[1)]	Int

1) ≥V8.0

Parameters not mapped directly to technology object data block tags

For measuring input technology objects, the following additional parameters are available.

Name in Openness	Name in function view	Possible value	Data type in Openness
_AssociatedObject	Associated object	PublicAPI-object	SW.TechnologicalObjects.TechnologicalInstanceDB
_ListenToMeasuringInput	Listen to measuring input	Technology objects of type TO_MeasuringInput	SW.TechnologicalObjects.TechnologicalInstanceDB

Output cam and cam track

Parameters mapped directly to technology object data block tags

For output cam technology objects the following read-only data block tags are writable in the PublicAPI.

Name in Openness	Name in function view	Possible value	Data type in Openness
Interface.LogicOperation	Logical operation of the output cam signal at the output	0: Logical OR 1: Logical AND	int

For camtrack technology object, no read-only data block tags are writable in the PublicAPI.

Parameters not mapped directly to technology object data block tags

For output cam and cam track technology objects, the following additional parameters are available.

Name in Openness	Name in function view	Possible value	Data type in Openness
_AssociatedObject	Associated object	PublicAPI-object	SW.TechnologicalObjects.TechnologicalInstanceDB

Kinematics

Parameters mapped directly to technology object data block tags

For kinematics technology object, the following read-only data block tags are writable in the PublicAPI.

Name in Openness	Name in function view	Possible value	Data type in Openness
Kinematics.TypeOfKinematics	Kinematics type	See tag Kinematics.TypeOfKinematics[1)]	int
MotionQueue.MaxNumberOfCommands	Maximum number of jobs	1..10	Int

[1)] Possible values are described in the version specific function manual S7-1500T Kinematics functions on chapter "Kinematics tags".

Parameters not mapped directly to technology object data block tags

For kinematics technology object, the following additional parameters are available:

Name in Openness	Name in function view	Possible value	Data type
_KinematicsAxis[n][1)]	Kinematics axis A1 .. A6	Axis that can be connected to TO_Kinematics objects	SW.TechnologicalObjects.TechnologicalInstanceDB
_Units.LengthUnit	Units of measurement > Position	See tag Units.LengthUnit[2)]	uint

Name in Openness	Name in function view	Possible value	Data type
_Units.LengthVelocityUnit	Units of measurement > Velocity	See tag Units.LengthVelocityUnit[2)]	uint
_Units.AngleUnit	Units of measurement > Angle	See tag UnitsAngleUnit[2)]	uint
_Units.AngleVelocityUnit	Units of measurement > Angle velocity	See tag Units.AngleVelocityUnit[2)]	uint
_Properties.UseHighResolutionPositionValues[1)]	Use position values with higher resolution	True, False	bool
_X_Minimum[3)]	x minimum	-1.00E+12 .. +1.00E+12	double
_X_Maximum[3)]	x maximum	-1.00E+12 .. +1.00E+12	double
_Y_Minimum[3)]	y minimum	-1.00E+12 .. +1.00E+12	double
_Y_Maximum[3)]	y maximum	-1.00E+12 .. +1.00E+12	double
_Z_Minimum[3)]	z minimum	-1.00E+12 .. +1.00E+12	double
_Z_Maximum[3)]	z maximum	-1.00E+12 .. +1.00E+12	double
_A3_Maximum[3)]	A3 maximum	-1.00E+12 .. +1.00E+12	double

1) <V7.0 n=1≤n≤4; ≥V7.0 1≤n≤6; Kinematics axes A5 and A6 can only be used with "S7-1500T Motion Control KinPlus".

2) Possible values are described in the version specific function manual S7-1500T Kinematics functions on chapter "Units tags".

3) ≥V5.0

Interpreter

Parameters mapped directly to technology object data block tags

For Interpreter technology object, the following read-only data block tags are writable in the PublicAPI.

Name in Openness	Name in function view	Possible value	Data type in Openness
Parameter.MaxNumberOfCommands	Maximum number of jobs	10..100	Int

Connecting drives

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- An S7-1500 PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266).
- A technology object of the type TO_SpeedAxis, TO_PositioningAxis or TO_SynchronousAxis is determined in the project.
- A drive is determined in the project.

Application

To connect an axis with a drive, it is necessary to specify several values together in a single call.

Sensor interface and Telegram 750

- If the actor interface contains a sensor interface, the sensor interface is connected automatically by using these methods.
- If the actor interface contains torque data, the additional telegramm 750 is connected automatically by using these methods.
- To access the sensor interface you can use SensorInterface[m] with 0≤m≤3.

The public API type AxisEncoderHardwareConnectionInterface provides the following methods which can be used to connect and disconnect the actor or sensor interfaces:

Method	Description
void Connect(HW.DeviceItem moduleInOut)	Connects to input and output addresses at the same module. To connect drives that are configured by using a GSD file use this method.
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut)	Connects to input and output addresses at separate modules. To connect drives that are configured by using a GSD file use this method.
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut, ConnectOption connectOption)	Connects to input and output addresses at separate modules, specifying an additional ConnectOption. To connect drives that are configured by using a GSD file use this method.
void Connect(HW.Channel channel)	Connects to a channel To connect technology modules use this method.
void Connect(MC.Drives.Telegram telegram)	Connects to a telegram provided by StartDrive drive object To connect drives that are configured by Startdrive as of V19 use this method.
void Connect(MC.Drives.Telegram telegram, ConnectOption connectOption)	Connects to a telegram provided by StartDrive drive object, specifying an additional ConnectOption. To connect drives that are configured by Startdrive as of V19 use this method.
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Connects specifying bit addresses directly You can use this method in any cases. If the drive is configured with Startdrive < V19, you must use this method. Read the Byte address of the telegram from the hardware configuration and use the corresponding Bit address. Bit address = Byte address * 8
void Connect(string pathToDBMember)	Connects to a data block tag To connect an axis to a drive via data block use this method. The data block for the connection must exist.
void Connect(SW.Tags.PlcTag outputTag)	Connects to a PLC tag To connect a drive with analog setpoint interface to tags use this method. You can connect tags for "analog output", "enable output", and "ready input".
void Disconnect()	Disconnects an existing connection

ET200SP PTO 2

If you connect an ET200SP PTO 2 module to a drive by specifying bit addresses directly, you must use the following offsets:

Channel	Telegram	Bit offset to InputAddress	Bit offset to OutputAddress
0	1	0	0
	81	32	32
1	1	128	64
	81	160	96

Attribute

To determine how the technology object is connected you can use the following read-only attributes. The respective connection values are set only if the connection of the specific kind exists.

Attribute	Data type	Description
IsConnected	bool	TRUE: Interface is connected FALSE: Interface is not connected
InputOutputModule	HW.DeviceItem	Connected module that contains input and output addresses
InputModule	HW.DeviceItem	Connected module that contains input addresses The value is also set in case of an existing connection to a module containing input and output addresses.
OutputModule	HW.DeviceItem	Connected module that contains output addresses The value is also set in case of an existing connection to a module containing input and output addresses.
InputAddress	int	Logical input address of connected object, for example 256.
OutputAddress	int	Logical output address of connected object, for example 256.
ConnectOption	ConnectOption	Value of the ConnectOption that has been set when the connection was made: • Default Only modules that are recognized as valid connection partners can be selected. • AllowAllModules This option corresponds to selecting "Show all modules" in the user interface.
Channel	HW.Channel	Connected channel
Telegram	MC.Drives.Telegram	Connected telegram of drive object
PathToDBMember	string	Connected technology object data block tag
OutputTag	SW.Tags.PlcTag	Connected PLC tag (analog connection)
SensorIndexInActorTelegram	int	Connected sensor interface in actor telegram The attribute is only relevant for sensor interfaces. 0: Encoder is not connected 1: Encoder is connected to first sensor interface in telegram 2: Encoder is connected to second sensor interface in telegram For the actor interface the value is always 0.

Program code to connect a mixed module with input and output addresses at the same module

Modify the following program code to connect a servo drive object from GSDML drive with input and output addresses at the same module:

```
public void Connect_ModuleInOut()
{
   var deviceItem = GsdmlDevice.DeviceItems.First(x => x.Name == "SIEMENS telegram 105,
PZD-10/10");
   var syncAxis =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find("SynchronousAxis_1");
   syncAxis
    .GetService<AxisHardwareConnectionProvider>()
    .ActorInterface
    .Connect(deviceItem);
}
```

Program code to connect input and output addresses at separate modules

Modify the following program code to connect a servo drive object from GSDML drives with input and output addresses at different modules:

```
//"Input module", "Output module" and "syncAxis" must be replaced with the names in your
project.
public void Connect_ModuleIn_ModuleOut()
{
    var deviceItemOut = Plc.DeviceItems.First(di => di.Name == "Input
module").DeviceItems[0];
    var deviceItemIn = Plc.DeviceItems.First(di => di.Name == "Output
module").DeviceItems[0];
    var syncAxis =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find("SynchronousAxis_1");
    syncAxis
    .GetService<AxisHardwareConnectionProvider>()
    .ActorInterface
    .Connect(deviceItemIn, deviceItemOut, ConnectOption.AllowAllModules);
}
```

If TIA Portal does not detect the module as a standard telegram, the connectOption AllowAllModules must be used.

Program code to connect technology modules

Modify the following program code to connect a technology module:

```
public void Connect_Channel()
{
   var tmTimer = Plc.DeviceItems.First(di => di.Name== "TM Timer DIDQ
16x24V_1").DeviceItems[0];
   var syncAxis =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find("SynchronousAxis_1");
    syncAxis.GetService<OutputCamMeasuringInputContainer>()
    .OutputCams.Find("OutputCam_1")
    .GetService<OutputCamHardwareConnectionProvider>()
    .Connect(tmTimer.Channels[0]);
}
```

Program code to connect drives configured via Startdrive as of V19

Startdrive as of V19 must be installed.

Modify the following program code to connect a servo drive object from Startdrive by telegram:

```
public void Connect_To_Startdrive(PlcSoftware plcsoftware)
{
  var mainTelegram = StartdriveDevice.DeviceItems.First(di => di.PositionNumber == 0)
   .GetService<DriveObjectContainer>().DriveObjects[0].Telegrams.Find(TelegramType.MainTele
gram);
  var syncAxis =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find("SynchronousAxis_1");

syncAxis.GetService<AxisHardwareConnectionProvider>().ActorInterface.Connect(mainTelegram)
;
}
```

Program code to connect drives configured via Startdrive < V19

Startdrive must be installed.

Modify the following program code to connect a servo drive object from StartDrive:

```
public void Connect_To_Startdrive()
{
  var mainTelegram = StartdriveDevice.DeviceItems.First(di => di.PositionNumber == 0)
  .GetService<DriveObjectContainer>().DriveObjects[0].Telegrams.Find(TelegramType.MainTele
gram);
  var syncAxis =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find("SynchronousAxis_1");
  var inputAddress = mainTelegram.Addresses.First(x => x.IoType ==
AddressIoType.Input).StartAddress;
  var outputAddress = mainTelegram.Addresses.First(x => x.IoType ==
AddressIoType.Output).StartAddress;
  syncAxis
   .GetService<AxisHardwareConnectionProvider>()
   .ActorInterface
//Byte adresses must be multiplied by 8 to get Bit addresses
   .Connect(inputAddress * 8, outputAddress * 8, ConnectOption.Default);
}
```

Program Code to connect data block tags

In this example the sensor of an external encoder is connected to the data block named "InstDBTel83". InstDBTel83 contains one member "Telegram" of typ "PD_Tel83":

Modify the following program code to connect to a data block tag.

```
public void Connect_PathToDBMember()
{
   var extEnc =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find("ExternalEncoder_1");
   extEnc
    GetService<EncoderHardwareConnectionProvider>()
    .SensorInterface
    .Connect("InstDBTel83.Telegram.Sensor_1");
}
```

Connecting telegram 750

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- An S7-1500 PLC is determined in the project.

- A technology object of the type TO_SpeedAxis, TO_PositioningAxis or TO_SynchronousAxis V4.0 is determined in the project.
- A drive that supports telegram 750 is determined in the project.

Application

If telegram 750 was added after connecting the drive and the axis, it is necessary to connect telegram 750 separately. EnableTorqueData is set to TRUE automatically. The public API type TorqueHardwareConnectionInterface provides the following methods which can be used to connect and disconnect telegram 750:

Method	Description
void Connect(HW.DeviceItem moduleInOut)	Connects to input and output addresses at the same module. To connect telegram 750 for drives that are configured by using a GSD file use this method.
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut)	Connects to input and output addresses at separate modules.
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut, ConnectOption connectOption)	Connects to input and output addresses at separate modules, specifying an additional ConnectOption.
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Connects specifying bit addresses directly. You can use this method in any cases. If the drive is configured with StartDrive, you must use this method. Read the Byte address of the telegram from the hardware configuration and use the corresponding Bit address. Bit address = Byte address * 8
void Connect(string pathToDBMember)	Connects to a data block tag. To connect telegram 750 via data block use this method. The data block for the connection must exist.
void Disconnect()	Disconnects an existing connection.

The TorqueHardwareConnectionInterface can be retrieved via the property TorqueInterface at the type AxisHardwareConnectionProvider. If the connection to telegram 750 is not supported, the property value is “null”.

If the drive is connected by data block tags, you cannot connect telegram 750 by module. You can use the following read-only attributes to determine how the technology object is

connected. The respective connection values are set only if the connection of the specific kind exists:

Attribute	Data type	Description
IsConnected	bool	TRUE: Interface is connected FALSE: Interface is not connected
InputOutput-Module	HW.DeviceItem	Connected module that contains input and output addresses
InputModule	HW.DeviceItem	Connected module that contains input addresses The value is also set in case of an existing connection to a module containing input and output addresses.
OutputModule	HW.DeviceItem	Connected module that contains output addresses The value is also set in case of an existing connection to a module containing input and output addresses.
InputAddress	int	Logical input address of connected object, for example 256
OutputAddress	int	Logical output address of connected object, for example 256
ConnectOption	ConnectOption	Value of the ConnectOption that has been set when the connection was made: • Default Only modules that are recognized as valid connection partners can be selected. • AllowAllModules This option corresponds to selecting "Show all modules" in the user interface.
PathToDB-Member	string	Connected technology object data block tag

Program Code: Connect telegramm 750

Modify the following program code to connect a mixed module that contains input and output addresses:

```
//An instance of technology object and device item is already available in the program
before
private static void ConnectTorqueInterface(TechnologicalInstanceDB technologyObject,
DeviceItem devItem)
{
    //Retrieve service AxisHardwareConnectionProvider
    AxisHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<AxisHardwareConnectionProvider>();
    //Connect TorqueInterface with DeviceItem
    connectionProvider.TorqueInterface.Connect(devItem);
}
```

Connecting encoders

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- An S7-1500 PLC is determined in the project.
- A technology object of the type TO_ExternalEncoder is determined in the project.
- An object is determined in the project that provides PROFIdrive telegram 81 or 83.

Application

To connect an external encoder technology object with the encoder hardware, it is necessary to specify several values together in a single call. The public API type AxisEncoderHardwareConnectionInterface provides the following methods which can be used to connect and disconnect the sensor interface:

Method	Description
void Connect(HW.DeviceItem moduleInOut)	Connects to input and output addresses at the same module. To connect encoders that are configured by using a GSD file use this method.
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut)	Connects to input and output addresses at separate modules. To connect encoders that are configured by using a GSD file use this method.
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut, ConnectOption connectOption)	Connects to input and output addresses at separate modules, specifying an additional ConnectOption To connect encoders that are configured by using a GSD file use this method.
void Connect(HW.Channel channel)	Connects to a channel. To connect technology modules use this method.
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Connects specifying bit addresses directly. You can use this method in any cases. If the encoder is configured with StartDrive, you must use this method. Read the Byte address of the telegram from the hardware configuration and use the corresponding Bit address. Bit address = Byte address * 8
void Connect(string pathToDBMember)	Connects to a data block tag. To connect an external encoder via data block use this method. The data block for the connection must exist.
void Connect(SW.Tags.PlcTag outputTag)	Not relevant for connecting encoders
void Disconnect()	Disconnects an existing connection.

You can use the following read-only attributes to determine how the technology object is connected. The respective connection values are set only if the connection of the specific kind exists.

Attribute	Data type	Description
IsConnected	bool	TRUE: Interface is connected FALSE: Interface is not connected
InputOutputModule	HW.DeviceItem	Connected module that contains input and output addresses
InputModule	HW.DeviceItem	Connected module that contains input addresses The value is also set in case of an existing connection to a module containing input and output addresses.
OutputModule	HW.DeviceItem	Connected module that contains output addresses The value is also set in case of an existing connection to a module containing input and output addresses.
InputAddress	int	Logical input address of connected object, for example 256.
OutputAddress	int	Logical output address of connected object, for example 256.
ConnectOption	ConnectOption	Value of the ConnectOption that has been set when the connection was made: • Default Only modules that are recognized as valid connection partners can be selected. • AllowAllModules This option corresponds to selecting "Show all modules" in the user interface.
Channel	HW.Channel	Connected channel
PathToDBMember	string	Connected data block tag
OutputTag	SW.Tags.PlcTag	Not relevant for connecting encoders
SensorIndexInActorTelegram	int	Connected sensor telegram The attribute is only relevant for sensor interfaces. 0: Encoder is not connected 1: Encoder is connected to first sensor interface in telegram 2: Encoder is connected to second sensor interface in telegram For the actor interface the value is always 0.

Program code: Connect an encoder

Modify the following program code to connect an external encoder technology object:

```
//An instance of technology object and device item is already available in the program
before
private static void UseServiceEncoderHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, DeviceItem devItem)
{
    //Retrieve service EncoderHardwareConnectionProvider
    EncoderHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<EncoderHardwareConnectionProvider>();

    //Connect SensorInterface with DeviceItem
    connectionProvider.SensorInterface.Connect(devItem);

    //Check ConnectionState of SensorInterface
    bool sensorInterfaceConnectionState = connectionProvider.SensorInterface.IsConnected;
}
```

Connecting output cams and cam tracks to hardware

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_OutputCam or TO_CamTrack is determined in the project.
- A digital output module is determined in the project, for example TM Timer DIDQ.

Application

To connect an output cam or cam track technology object with a digital output, it is necessary to specify several values together in a single call. The public API type OutputCamHardwareConnectionProvider provides the following methods which can be used to connect and disconnect the actor or sensor interfaces:

Method	Description
void Connect(HW.Channel channel)	Connects to a channel
void Connect(SW.Tags.PlcTag outputTag)	Connects to a PLC tag
void Connect(int address)	Connects specifying bit addresses directly
void Disconnect()	Disconnects an existing connection

You can use the following read-only attributes to determine how the technology object is connected:

Attribute	Data type	Description
IsConnected	bool	TRUE: Technology object is connected FALSE: Technology object is not connected
Channel	HW.Channel	Connected channel
OutputTag	SW.Tags.PlcTag	Connected PLC tag
OutputAddress	int	Logical output address of connected object, for example 256.

Program code: Connect output cam or cam track technology object

Modify the following program code to connect an output cam or cam track technology object:

```
//An instance of technology object and channel item is already available in the program
before
private static void UseServiceOutputCamHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, Channel channel)

{
    //Retrieve service OutputCamHardwareConnectionProvider
    OutputCamHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<OutputCamHardwareConnectionProvider>();

    //Connect technology object with Channel
    connectionProvider.Connect(channel);

    //Check ConnectionState of technology object
    bool connectionState = connectionProvider.IsConnected;
}
```

Program Code to connect PLC tags

Modify the following program code to connect an output cam to a PLC tag. The PLC tag points to the output address:

```
public void Connect_PlcTag(PlcSoftware plcSoftware)
{
   var outputCamTag = plcSoftware.TagTableGroup.TagTables.Find("My Tag
Table").Tags.Find("OutputCamTag");
   var syncAxis =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find("SynchronousAxis_1");
   syncAxis.GetService<OutputCamMeasuringInputContainer>()
    .OutputCams.Find("OutputCam_1")
    .GetService<OutputCamHardwareConnectionProvider>()
    .Connect(outputCamTag);
}
```

Connecting measuring inputs to hardware

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_MeasuringInput is determined in the project.
- A digital input module is determined at drive or in the project, for example TM Timer DIDQ.

Application

To connect a measuring input technology object with a digital input, it is necessary to specify several values together in a single call. The public API type MeasuringInputHardwareConnectionProvider provides the following methods which can be used to connect and disconnect the actor or sensor interface:

Method	Description
void Connect(HW.Channel channel)	Connects to a channel
void Connect(HW.DeviceItem moduleIn, int channelIndex)	Connects to a module, specifying an additional channel index
void Connect(int address)	Connects specifying bit addresses directly • For TM Channels, the bit address can be calculated via the formula bit_address = module_start_address * 8 + channel_index • For telegram 39x, the associated formula is bit_address = module_start_address * 8 + 24 + number_of_measuringinput_in_telegram
void Disconnect()	Disconnects an existing connection

You can use the following read-only attributes to determine how the technology object is connected:

Attribute	Data type	Description
IsConnected	bool	TRUE: Technology object is connected FALSE: Technology object is not connected
InputModule	HW.DeviceItem	Connected module that contains input addresses
ChannelIndex	int	Index of connected channel with respect to InputModule
Channel	HW.Channel	Connected channel
InputAddress	int	Logical input address of connected object, for example 256.

Program code: Connect a measuring input technology object

Modify the following program code to connect a measuring input technology object:

```
//An instance of technology object and channel item is already available in the program
before
private static void
UseServiceMeasuringInputHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, Channel channel)
{
    //Retrieve service MeasuringInputHardwareConnectionProvider
    MeasuringInputHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<MeasuringInputHardwareConnectionProvider>();

    //Connect technology object with Channel
    connectionProvider.Connect(channel);

    //Check ConnectionState of technology object
    bool connectionState = connectionProvider.IsConnected;
}
```

Connecting synchronous axis with leading values

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_PositioningAxis, TO_SynchronousAxis, TO_ExternalEncoder or TO_LeadingAxisProxy as leading axis is determined in the project.
- A technology object of the type TO_SynchronousAxis as following axis is determined in the project.

Application

To connect a synchronous axis technology object with leading values, it is necessary to specify several values together in a single call. The public API type SynchronousAxisMasterValues provides the following methods which can be used to connect and disconnect leading values. Leading values can be connected as setpoint coupling (S7-1500 PLC, S7-1500T PLC) or actual value coupling (S7-1500T PLC). All methods and attributes are relevant for both types of coupling.

Method	Description
int IndexOf (TechnologicalInstanceDB element)	Returns the corresponding index of a leading value
bool Contains (TechnologicalInstanceDB element)	TRUE: The container contains the leading value FALSE: The container does not contain the leading value

Method	Description
IEnumerator GetEnumerator <TechnologicalInstanceDB>()	Used to support each iteration
void Add (TechnologicalInstanceDB element)	Connects following axis to leading value
bool Remove (TechnologicalInstanceDB element)	Disconnects following axis from leading value TRUE: Disconnection was succesfully FALSE: Disconnection was not succesfully

You can use the following read-only attributes:

Attribute	Data type	Description
Count	int	Count of leading values
IsReadonly	bool	TRUE: The container is read-only FALSE: The container is not read-only
Parent	IEngineeringObject	Returns the parent of the container. In this case parent means the service SynchronousAxisMasterValues.
this [id] { get; }	TechnologicalInstanceDB	Index-based access to leading values

You can use the following attributes to connect a synchronous axis with a leading value:

Attribute	Data type	Description
SetPointCoupling	SW.TechnologicalObjects.TechnologicalInstanceDBAssociation	The attribute gets container of connected leading axis with setpoint coupling.
ActualValueCoupling	SW.TechnologicalObjects.TechnologicalInstanceDBAssociation	The attribute gets container of connected leading axis with actual value coupling.
DelayedCoupling	SW.TechnologicalObjects.TechnologicalInstanceDBAssociation	The attribute contains all master values that are coupled via delayed values.

Program code: Connect a synchronous axis with a leading value

Modify the following program code to connect a synchronous axis with a leading value:

```
//An instance of leading axis and following axis is already available in the program before
private static void UseServiceSynchronousAxisMasterValues(TechnologicalInstanceDB
masterTechnologyObject, TechnologicalInstanceDB synchronousTechnologyObject)
{
    //Retrieve service SynchronousAxisMasterValues
    SynchronousAxisMasterValues masterValues =
    synchronousTechnologyObject.GetService<SynchronousAxisMasterValues>();

    //Connect following axis and leading axis with setpoint coupling
    masterValues.SetPointCoupling.Add(masterTechnologyObject);

    //Get container of connected leading axis with setpoint coupling
    TechnologicalInstanceDBAssociation setPointMasterValues =
    masterValues.SetPointCoupling;

    //Remove connected leading axis with setpoint coupling
    masterValues.SetPointCoupling.Remove(masterTechnologyObject);

    //Connect following axis and leading axis with actual value coupling
    masterValues.ActualValueCoupling.Add(masterTechnologyObject);

    //Get container of connected leading axis with actual value coupling
    TechnologicalInstanceDBAssociation actualValueMasterValues =
    masterValues.ActualValueCoupling;

    //Remove connected leading axis with actual value coupling
    masterValues.ActualValueCoupling.Remove(masterTechnologyObject);
}
```

Connecting conveyor tracking of kinematics with leading values

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- A S7-1500T PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266).
- A technology object of the type TO_PositioningAxis, TO_SynchronousAxis or TO_ExternalEncoder as leading axis is determined in the project.
- A technology object of the type TO_Kinematics as following axis is determined in the project.

Application

To configure leading values for conveyor tracking, use the service SW.TechnologicalObjects.Motion.ConveyorTrackingLeadingValues for TO Kinematics ≥ V5.0.

Method	Description
int IndexOf (TechnologicalInstanceDB element)	Returns the corresponding index of a leading value
bool Contains (TechnologicalInstanceDB element)	TRUE: The container contains the leading value FALSE: The container does not contain the leading value
IEnumerator GetEnumerator <TechnologicalInstanceDB>()	Used to support each iteration
void Add (TechnologicalInstanceDB element)	Connects following axis to leading value
bool Remove (TechnologicalInstanceDB element)	Disconnects following axis from leading value TRUE: Disconnection was succesfully FALSE: Disconnection was not succesfully

You can use the following read-only attributes:

Attribute	Data type	Description
Count	int	Count of leading values
IsReadonly	bool	TRUE: The container is read-only FALSE: The container is not read-only
Parent	IEngineeringObject	Returns the parent of the container. In this case parent means the service SynchronousAxisMasterValues.
this [id] { get; }	TechnologicalInstanceDB	Index-based access to leading values

You can use the following attributes to connect a kinematics axis with a leading value:

Attribute	Data type	Description
SetPointCoupling	SW.TechnologicalObjects.TechnologicalInstanceDBAssociation	The attribute gets container of connected leading axis with setpoint coupling.
ActualValueCoupling	SW.TechnologicalObjects.TechnologicalInstanceDBAssociation	The attribute gets container of connected leading axis with actual value coupling.
DelayedCoupling	SW.TechnologicalObjects.TechnologicalInstanceDBAssociation	The attribute contains all master values that are coupled via delayed values.

Connect conveyor tracking axis of a kinematics with a leading value

An example of the interconnection is included in the topic Connecting synchronous axis with leading values (Page 571).

Exporting and importing technology object cam (S7-1500T)

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- An S7-1500 PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266)
- The technology object exists.

Application

To export or import the data of a technology object cam of typ TO_Cam or TO_Cam_10k you have to specify the format and which separator should be used. The public API type CamDataSupport provides the following methods which can be used to export the data of technology object cam. For more information on export format of cam, Refer function manual Using S7-1500/S7-1500T Synchronous Operations Functions on chapter Importing / exporting cam.

Method	Description
void SaveCamDataBinary(System.IO.FileInfo destinationFile)	Saves the cam data in binary format in the destination file.
void SaveCamDataPointList(System.IO.FileInfo destinationFile, CamDataFormatSeparator separator, int samplePoints)	Saves the cam data in format "PointList" in the destination file.
void SaveCamData(System.IO.FileInfo destinationFile, CamDataFormat format, CamDataFormatSeparator separator)	Saves the cam data in the destination file. You can specify data format as "MCD", "SCOUT" or "Pointlist" and separator as "tab" or "comma". If you choose "PointList" 360 interpolation points will be exported.
void LoadCamData(System.IO.FileInfo sourceFile, CamDataFormatSeparator separator)	Loads the cam data in the format "MCD", "SCOUT" or "Pointlist" to the project.
void LoadCamDataBinary(System.IO.FileInfo sourceFile)	Loads the cam data from a binary file to the project.

You can use the following attributes:

Attribute	Data type	Description
separator	CamDataFormatSeparator	Allowed values • tab • comma
samplePoints	int	Number of interpolation points that should be exported.
format	CamDataFormat	Allowed values • MCD • SCOUT • Pointlist

Attribute	Data type	Description
destinationFile	System.IO.FileInfo	Name of target file. Must not be null. Access rights and enough space on storage medium must be given. An existing file will be overwritten.
sourceFile	System.IO.FileInfo	Name of source file. Must not be null. Access rights must be given. Content must be in specified format.

Program code: Save cam data

Modify the following program code to save cam data:

```
//An instance of technology object is already available in the program before
private static void ExportCamData(TechnologicalInstanceDB technologyObject,
System.IO.FileInfo destinationFile)
{
    //Retrieve service CamDataSupport
    CamDataSupport camData = technologyObject.GetService<CamDataSupport>();

    //Save cam data in MCD format, using the separator Tab
    camData.SaveCamData(destinationFile, CamDataFormat.MCD, CamDataFormatSeparator.Tab);
}
```

Program code: Load cam data

Modify the following program code to load cam data:

```
//An instance of technology object is already available in the program before
private static void ImportCamData(TechnologicalInstanceDB technologyObject,
System.IO.FileInfo sourceFile)
{
    //Retrieve service CamDataSupport
    CamDataSupport camData = technologyObject.GetService<CamDataSupport>();

    //Load cam data from source file, using the separator Tab
    camData.LoadCamData(sourceFile, CamDataFormatSeparator.Tab);
}
```

Load and save the source code of an interpreter program (S71500T)

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).

- A S7-1500T PLC is determined in the project.
  See Querying PLC and HMI targets (Page 266).
- A technology object of the type TO_InterpeterProgram is determined in the project.

Application

The public API type InterpreterProgramSupport provides the following methods which can be used to load the MCL program from a text file into a TO_InterpreterProgram technology object or save it in a text file.

The MCL program is provided by a path to the source file.

Method	Description
void SaveSource(System.IO.FileInfo fileInfo)	Saves the MCL program from the TO_InterpreterProgram technology object to the specified txt-file.
void LoadSource(System.IO.FileInfo fileInfo)	Loads the specified txt-file with the MCL program into the TO_InterpreterProgram technology object.

Program code to save an MCL program

Modify the following program code to connect a technology module:

```
//An instance of technology object is already available in the program before
private static void ExportIprProgramSourceCode(TechnologicalInstanceDB technologyObject,
System.IO.FileInfo destinationFile)
{
   //Retrieve service InterpreterProgramSupport
    InterpreterProgramSupport iprProgramSourceCode =
technologyObject.GetService<InterpreterProgramSupport>();

//Save source code in the destination file
    iprProgramSourceCode.SaveSource(destinationFile);
}
```

Program code to load an MCL program

Modify the following program code to connect a technology module:

```
//An instance of technology object is already available in the program before
private static void ImportIprProgramSourceCode(TechnologicalInstanceDB technologyObject,
System.IO.FileInfo sourceFile)
{
   //Retrieve service InterpreterProgramSupport
    InterpreterProgramSupport iprProgramSourceCode =
technologyObject.GetService<InterpreterProgramSupport>();

    //Load source code from source file
    iprProgramSourceCode.LoadSource(sourceFile);
}
```

5.11.4.19 PID control

Parameters for PID_Compact, PID_3Step, PID_Temp, CONT_C, CONT_S, TCONT_CP and TCONT_S

You can find a list of all available variables in SIMATIC S7-1200/S7-1500 PID control function manual on the internet (https://support.industry.siemens.com/cs/ww/en/view/108210036).

Note

In TIA Portal in the Parameter view of the technology object configuration you can find the column "Name in Openness".

5.11.4.20 Counting

Parameters for High_Speed_Counter and SSI_Absolute_Encoder

You can find a list of all available parameters in the product information “Parameters of technology objects in TIA Portal Openness“ on the internet (https://support.industry.siemens.com/cs/ww/en/view/109744932).

For each parameter the following properties are provided:

- Name in configuration (TIA Portal)
- Name in Openness
- Data type in Openness
- Default access
- Range of values

Additional information

You can find additional information in SIMATIC S7-1500, ET 200MP, ET 200SP Counting, measurement and position input function manual on the internet (http://support.automation.siemens.com/WW/view/en/59709820).

5.11.4.21 Easy Motion Control

Parameters for AXIS_REF

You can find a list of all available parameters in the product information “Parameters of technology objects in TIA Portal Openness“ on the internet (https://support.industry.siemens.com/cs/ww/en/view/109744932).

For each parameter the following properties are provided:

- Name in configuration (TIA Portal)
- Name in Openness

- Data type in Openness
- Default access
- Range of values

Note

In TIA Portal in the Parameter view of the technology object configuration you can find the column "Name in Openness".

Additional information

You can find additional information for Easy Motion Control in the Information system of STEP 7 (TIA Portal).

5.11.5 Tags and Tag tables

5.11.5.1 Starting the "PLC Tags" editor

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- Instance of the TIA Portal is opened with user interface.

Program code

Modify the following program code to start the corresponding editor for an object reference of the type `PlcTagTable` in the TIA Portal instance:

```
//Opens tagtable in editor "Tags"
private static void OpenTagtableInEditor(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
plcTagTable.ShowInEditor();
}
```

See also

Importing configuration data (Page 1212)

5.11.5.2 Querying system groups for PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PlcSoftware instance was retrieved from a PLC device item.
  See Querying PLC and HMI targets (Page 266)

Program code

Modify the following program code to query the system group for PLC tags:

```
//Retrieves the plc tag table group from a plc
private PlcTagTableSystemGroup GetControllerTagfolder(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    return plcTagTableSystemGroup;
}
```

5.11.5.3 Creating PLC tag table

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PlcSoftware instance was retrieved from a PLC device item.
  See Querying PLC and HMI targets (Page 266)

Program code

Modify the following program code to create the PLC tag table. It creates a new tag table with the given name in the composition.

```
PlcTagTable myTable = plc.TagTableGroup.TagTables.Create("myTable");
```

See also

Querying PLC and HMI targets (Page 266)

5.11.5.4 Enumerating user-defined groups for PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A PlcSoftware instance was retrieved from a PLC device item.
  See Querying PLC and HMI targets (Page 266)

Application

Subfolders are taken into account recursively for enumeration.

Program code: Enumerating user-defined groups for PLC tags

Modify the following program code to enumerate user-defined groups for PLC tags:

```
//Enumerates all plc tag table user groups including subgroups
private static void EnumeratePlcTagTableUserGroups(PlcSoftware plcSoftware)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in plcSoftware.TagTableGroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
    }
}
private static void EnumerateTagTableUserGroups(PlcTagTableUserGroup tagTableUsergroup)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in tagTableUsergroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
        // recursion
    }
}
```

Program code: Accessing a user-defined group

Modify the following program code to access a user-defined group for PLC tags:

```
//Gives individual access to a specific plc tag table user folder
private static void AccessPlcTagTableUserGroupWithFind(PlcSoftware plcSoftware, string
folderToFind)
{
    PlcTagTableUserGroupComposition plcTagTableUserGroupComposition =
plcSoftware.TagTableGroup.Groups;
    PlcTagTableUserGroup controllerTagUserFolder =
plcTagTableUserGroupComposition.Find(folderToFind);
    // The parameter specifies the name of the user folder
}
```

5.11.5.5 Creating user-defined groups for PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The TIA Portal Openness API interface supports the creation of a user-defined group for PLC tags.

Program code

Modify the following program code to create a user-defined group for PLC tags:

```
//Creates a plc tag table user group
private static void CreatePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup systemGroup = plcSoftware.TagTableGroup;
    PlcTagTableUserGroupComposition groupComposition = systemGroup.Groups;
    PlcTagTableUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
    // Optional;
    // create a subgroup
    PlcTagTableUserGroup mySubCreatedGroup =
myCreatedGroup.Groups.Create("MySubSubGroupName");
}
```

5.11.5.6 Deleting user-defined groups for PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The TIA Portal Openness API interface supports the deletion of a specific user-defined group for PLC tag tables.

Program code

Modify the following program code to delete a specific user-defined group for PLC tag tables:

```
private static void DeletePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableUserGroup group = plcSoftware.TagTableGroup.Groups.Find("MySubGroupName");
    if (group != null)
    {
        group.Delete();
    }
}
```

5.11.5.7 Enumerating PLC tag tables in a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code: Enumerating PLC tag tables

Modify the following program code to enumerate all PLC tag tables in system groups or in user-defined groups:

```
//Enumerates all plc tag tables in a specific system group or and user group
private static void EnumerateAllPlcTagTablesInFolder(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    foreach (PlcTagTable tagTable in tagTables)
    {
        // add code here
    }
}
```

Program code: Accessing PLC tag table

Modify the following program code to access the PLC tag table:

```
//Gives individual access to a specific Plc tag table
private static void AccessToPlcTagTableWithFind(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    PlcTagTable controllerTagTable = tagTables.Find("Tag table XYZ");
    // The parameter specifies the name of the tag table
}
```

5.11.5.8 Querying information from a PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Via PLC tag tables you can access user constants, system constants and tags. The count of the tag composition of a tag table is equal to the number of tags in that tag table. The PLCTagTable contains the following navigators, attributes, and actions.

The following attributes are accessed in PLC tag table.

Name	Type	Type
IsDefault	bool	Read-only
ModifiedTimeStamp	DateTime	Read-only
Name	string	Read-only

The PLCTag table contains the following actions as given below.

Name	Return type	Description
Delete	void	Deletes the instance. Throws an exception if IsDefault is true.
Export	void	Exports the Simatic ML of a Plc tag table.
ShowInEditor	void	Shows the tag table in the Plc tag table editor.

Program code

Modify the following program code to query the information for a PLC tag table:

```
private static void AccessPlcConstantsUsingFind(PlcTagTable tagTable)
{
    PlcUserConstantComposition plcUserConstants = tagTable.UserConstants;
    PlcUserConstant plcUserConstant = plcUserConstants.Find("Constant XYZ");
    //PlcSystemConstantComposition plcSystemConstants = tagTable.SystemConstants;
    //PlcSystemConstant plcSystemConstant = plcSystemConstants.Find("Constant XYZ");
}
private static void EnumeratePlcTags(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    foreach (PlcTag plcTag in plcTags)
    {
        string name = plcTag.Name; string typeName = plcTag.DataTypeName;
        string logicalAddress = plcTag.LogicalAddress;
    }
}
private static void EnumeratePlcTagsUsingFind(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    PlcTag plcTag = plcTags.Find("Constant XYZ");
}
```

5.11.5.9 Reading the time of the last changes of a PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The format of the time stamp is UTC.

Program code

Modify the following program code to read the time stamp of a specific PLC tag table:

```
//Reads Time-Stamp of a plc Tag Table
private static void GetLastModificationDateOfTagtable(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
    DateTime modifiedTagTableTimeStamp = plcTagTable.ModifiedTimeStamp;
}
```

5.11.5.10 Deleting a PLC tag table from a group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to delete a specific tag table from a group:

```
//Deletes a PlcTagTable of a group
private static void DeletePlcTagTableInAGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup group = plcSoftware.TagTableGroup;
    PlcTagTable tagtable = group.TagTables.Find("MyTagTable");
    if (tagtable!= null)
    {
        tagtable.Delete();
    }
}
```

5.11.5.11 Enumerating PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code: Enumerating PLC tags in tag tables

Modify the following program code to enumerate all PLC tags in a tag table:

```
//Enumerates all plc tags in a specific tag table
private static void EnumerateAllPlcTagsInTagTable(PlcSoftware plcSoftware)
{
    PlcTagTable tagTable = plcSoftware.TagTableGroup.TagTables.Find("Tagtable XYZ");
    foreach (PlcTag tag in tagTable.Tags)
    {
        // add code here
    }
}
```

5.11.5.12 Accessing PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The type `PlcTagComposition` represents a collection of plc tags.

Program code: Accessing a specific PLC tag

Modify the following program code to access the required PLC tag. You have access to the following attributes:

- Name (read only)
- Data type name
- Logical address
- Comment
- ExternalAccessible
- ExternalVisible
- ExternalWritable
- IsSafety (read only)

```
//Gives individual access to a specific plc tag
private static void AccessPlcTag(PlcTagTable tagTable)
{
    PlcTag tag = tagTable.Tags.Find("Tag XYZ");
    // The parameter specifies the name of the tag
}
```

Program code: Creating tags

Modify the following program code:

```
private static void CreateTagInPLCTagtable(PlcSoftware plcsoftware)
// Create a tag in a tag table with default attributes
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Create(tagName);
}
```

Modify the following program code:

```
private static void CreateTagInPLCTagtable(PlcSoftware plcsoftware)
// Create a tag of data type bool and logical address not set
{
    string tagName = "MyTag";
    string dataType = "Bool";
    string logicalAddress ="";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Create(tagName, dataType, logicalAddress);
}
```

Program code: Deleting tags

Modify the following program code:

```
private static void DeleteTagFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single tag of a tag table
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Find(tagName);
    if (tag != null)
    {
        tag.Delete();
    }
}
```

5.11.5.13 Accessing PLC constants

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The type `PlcUserConstantComposition` represents a collection of plc user constants. You have access to the following attributes:

- Name (read only)
- Data type name
- Value

The type `PlcSystemConstantComposition` represents a collection of plc system constants. You have access to the following attributes:

- Name (read only)
- Data type name (read only)
- Value (read only)

Program code: Creating user constants

Modify the following program code:

```
private static void CreateUserConstantInPLCTagtable(PlcSoftware plcsoftware)
// Create a user constant in a tag table
{
string constantName = "MyConstant";
PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
PlcUserConstantComposition userConstantComposition = table.UserConstants;
PlcUserConstant userConstant = userConstantComposition.Create(constantName);
}
```

Program code: Deleting user constants

Modify the following program code:

```
private static void DeleteUserConstantFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single user constant of a tag table
{
PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
PlcUserConstantComposition userConstantComposition = table.UserConstants;
PlcUserConstant userConstant = userConstantComposition.Find("MyConstant");
if (userConstant != null)
{
userConstant.Delete();
}
}
```

Program code: Accessing system constants

Modify the following program code:

```
//Gives individual access to a specific system constant
private static void AccessSystemConstant(PlcTagTable tagTable)
{
PlcTag systemConstant = tagTable.SystemConstants.Find("Constant XYZ");
// The parameter specifies the name of the tag
}
```

See also

Creating user-defined groups for PLC tags (Page 582)

Deleting user-defined groups for PLC tags (Page 583)

Deleting a PLC tag table from a group (Page 586)

Accessing PLC tags (Page 587)

Starting the "PLC Tags" editor (Page 579)

Reading the time of the last changes of a PLC tag table (Page 586)

5.11.6 Functions for software units

5.11.6.1 Accessing software unit

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access the units. The units are important parts of the PLC's programming. Only a selected set of PLCs are supporting units therefore units are not modelled statically in the object model so they aren't browsable directly under the PLC software container.

The PlcUnitProvider makes units accessible and is accessible from the PlcSoftware through GetService( ). The PlcUnitProvider allows the access to PlcUnitSystemGroup via the UnitGroup navigator which contains the PlcUnitComposition with the general composition specific possibilities (iterate, Find, Create, Import etc. for PlcUnits).

The unit's representation (PlcUnit) contains the unit's own specific properties as Openness attributes.

You can access the below attribute and method for Openness unit properties:

Attribute Name	Type
Author	String
Name	String

Method Name	Return type
Export	void
Delete	void

Method Name	Return type
GetAttributes	IList<object>
SetAttributes	void

On the PlcUnit, You can iterate through the following objects:

- Comment: MultilingualText
- the BlockGroup (PlcBlockSystemGroup)
  - unit.BlockGroup
- TagTableGroup (PlcTagTableSystemGroup)
  - unit.TagTableGroup
- TypeGroup (PlcTypeSystemGroup)
  - unit.TypeGroup

Program code

```
private static PlcUnit AccessSoftwareUnit(Project project, string plcName = "PLC_1", string
plcUnitName = "Unit_1")
{
PlcUnit plcUint = null;
Device device = null;
DeviceItem deviceItem = null;
if (project == null)
{
Console.WriteLine("Project cannot be null");
return plcUint;
}
//Finding the plc
Console.WriteLine(String.Format("Iterate through {0} device(s)", project.Devices.Count));
foreach (Device checkDevice in project.Devices)
{
if (device == null)
{
Console.WriteLine(String.Format("Device: \"{0}\".", checkDevice.Name));
foreach (DeviceItem checkDeviceItem in checkDevice.DeviceItems)
{
if (device == null)
{
if (checkDeviceItem.Name == plcName)
{
device = checkDevice;
deviceItem = checkDeviceItem;
}
}
}
}
}
if (device == null)
{
Console.WriteLine("Device cannot be null");
return plcUint;
}
else
{
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<SoftwareContainer>();
PlcSoftware software = softwareContainer.Software as PlcSoftware;
//Find the SoftwareUnit
PlcUnitProvider plcUnitProvider = software.GetService<PlcUnitProvider>();
plcUnit = plcUnitProvider.UnitGroup.Units.Find(plcUnitName);
if (plcUnit != null)
{
//Do something
UnitAccessType att = readBlockAttribute(plcUnit);
IList<object> atts = readBlockAttributeList(plcUnit);
}
}
return plcUnit;
}
```

See also

Opening a project (Page 128)

5.11.6.2 Working with software unit

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

The units are important parts of the PLC's Programming and can be found on the PlcSoftware.

The unit provider is retrieved from PlcSoftware via GetService and the PlcUnitComposition can be retrieved from it.

You can perform the following elementary operations for units while using TIA Portal Openness:

- Creating units
- Deleting units
- Renaming units

Program code: Creating units

Modify the following program code to create units.

```
private static void CreateUnit(PlcUnitProvider provider, string plcUnitName)
{
PlcUnitComposition unitComposition = provider.UnitGroup.Units;
//Creating Unit
PlcUnit unit1 = unitComposition.Create(plcUnitName);
}
```

Error will occur if you want to create a unit with which a name already exists. A Recoverable Exception will be thrown with the following message: "Unit with name'Unit1' already exists.

```
PlcUnit unit1 = unitComposition.Create("Unit1");
...
...
try
{
PlcUnit unit2 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetInvocationException  e)
{
Console.WriteLine(e.Message);
}
```

Exception handling: Creating units

Error will occur if you provide the wrong name, for example starting with whitespace or containing invalid character or name is too long. A Recoverable Exception will be thrown with the following error message: "The value of attribute 'Name' contains an invalid character at Position 0."

```
private static void CreateUnit(PlcUnitProvider provider, string plcUnitName)
{
PlcUnitComposition unitComposition = provider.UnitGroup.Units;
//Creating Unit
try
{
PlcUnit unit1 = unitComposition.Create(plcUnitName);
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
}
```

Error will occur if you want to create a unit with which a name already exists. A Recoverable Exception will be thrown with the following message: "Unit with name'Unit1' already exists.

```
private static void CreateUnit(PlcUnitProvider provider, string plcUnitName)
{
PlcUnitComposition unitComposition = provider.UnitGroup.Units;
try
{
PlcUnit unit2 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetInvocationException  e)
{
Console.WriteLine(e.Message);
}
```

Program code: Deleting units

Modify the following program code to delete units.

```
private static void DeleteUnit(PlcUnitProvider unitProvider, string plcUnitName)
{
unitProvider.UnitGroup.Units.Find(plcUnitName).Delete();
}
```

You will encounter NullReferenceException in case of the Find( ) method from above example code does not provide a reference to a Unit.

Program code: Renaming units

You can set the attributes in several ways. It can be adjusted via value assignment or call of SetAttribute as well as SetAttributes. The Name property can be validated by GetAttribute or GetAttributes. To be able to use SetAttribute or GetAttribute the object should be casted to IEngineeringObject.

Modify the following program code to rename units.

```
private void RenameUnit(PlcUnitProvider provider)
{
PlcUnitComposition unitComposition = provider.UnitGroup.Units;
//Set Value
PlcUnit unit1 = unitComposition.Create("Unit1");
unit1.Name = "Unit1_new";
//Using SetAttributes():
var attrList = new List<KeyValuePair<string, object>>()
{
new KeyValuePair<string, object>("Name", "Unit2_new")
};
unit1.SetAttributes(attrList);
//Using SetAttribute()
PlcUnit unit2 = unitComposition.Create("Unit2");
IEngineeringObject unit = (IEngineeringObject)unit2;
unit.SetAttribute("Name", "Unit2_new");
}
```

Error will occur if you provide wrong name, for example starting with whitespace, or containing invalid character. A Recoverable Exception will be thrown with the error message: "The value of attribute 'Name' contains an invalid character at Position 0."

```
PlcUnit unit1 = unitComposition.Create("Unit_1");
try
{
unit1.Name = "Unit_1";
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
```

Error will occur if you want to rename a unit with a name which already exists. A recoverable exception will be thrown with the error message: "The property 'Name' has an invalid value: 'Unit1'. A software unit with the same name already exists."

```
PlcUnit unit1 = unitComposition.Create("Unit1");
PlcUnit unit2 = unitComposition.Create("Unit_2");
try
{
PlcUnit unit2 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetInvocationException  e)
{
Console.WriteLine(e.Message);
}
```

See also

Opening a project (Page 128)

5.11.6.3 Accessing software unit underlying objects

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access:

- The Program blocks system group, its contained groups and blocks recursively
- The PLC data types system group, its contained groups and data types recursively
- The PLC tags system group, its contained groups and tag tables recursively

Program code: Accessing program block and block groups

Modify the following program code to access the program block under the current PLC unit by retrieving the program block composition and iterating through its contained block:

```
private void AccessProgramBlock(PlcUnit m_PlcUnit)
{
PlcBlockComposition blockComposition = m_PlcUnit.BlockGroup.Blocks;
foreach (PlcBlock block in blockComposition)
{
//usage of block
}
}
```

Modify the following program code to access the program block group under the current PLC unit by retrieving the program block group composition, and iterating through its contained groups:

```
private void AccessProgramBlockGroup(PlcUnit m_plcUnit)
{
PlcBlockUserGroupComposition usergroupComposition = m_PlcUnit.BlockGroup.Groups;
foreach (PlcBlockUserGroup group in usergroupComposition)
{
PlcBlockComposition blockComposition = group.Blocks;
foreach (PlcBlock block in blockComposition)
{
//…
//usage of the block
//…
}
}
}
```

Program code: Accessing PLC data types and data type groups

Modify the following program code to access PLC data types under the current PLC unit by retrieving the PLC data type composition and iterating through its contained data types:

```
private void AccessPlcDatatypes(PlcUnit m_PlcUnit)
{

PlcTypeComposition typeComposition = m_PlcUnit.TypeGroup.Types;
foreach (PlcType type in typeComposition)
{
// …
// usage of the type
// …
}
}
```

Modify the following program code to access PLC data type group under the current PLC unit by retrieving the PLC data type composition and iterating through its contained groups:

```
private void AccessPlcDatatypeGroup(PlcUnit m_PlcUnit)
{
PlcTypeUserGroupComposition usergroupComposition = m_PlcUnit.TypeGroup.Groups;
foreach (PlcTypeUserGroup group in usergroupComposition)
{
PlcTypeCompostion typeComposition = group.Types;
foreach (PlcType type in typeComposition)
{
// …
// usage of the type
// …
}
}
}
```

Program code: Accessing PLC tag tables and tag table goups

Modify the following program code to access the PLC tag tables under the current PLC unit by retrieving the PLC tag table composition and iterating through its contained tag tables:

```
private void AccessPlcTagTables(PlcUnit m_PlcUnit)
{
PlcTagTableComposition tagtableComposition = m_PlcUnit.TagTableGroup.TagTables;
foreach (PlcTagTable type in tagtableComposition)
{
//…
// usage of the tag table
//…
}
}
```

Modify the following program code to access PLC tag table groups under the current PLC unit by retrieving the PLC tag table group composition and iterating through its contained groups:

```
private void AccessPlcTagTableGroups(PlcUnit m_PlcUnit)
{
PlcTagTableUserGroupComposition usergroupComposition = m_PlcUnit.TagTableGroup.Groups;
foreach (PlcTagTableUserGroup group in usergroupComposition)
{
PlcTagTableComposition tagtableComposition = group.TagTables;
foreach (PlcTagTable type in tagtableComposition)
{
//…
// usage of the tag table
//…
}
}
}
```

5.11.6.4 Accessing to existing relations of a unit

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access all the existing relations of a unit so that you can read relation properties.

The following attributes are supported by the unit relation in the Openness:

Attribute name	Type	description
RelationType	UnitRelationType	Type of the relation
RelatedObject	String	Contains the name of the accessible element

The following ENUM values are provided for the attribute RelationType:

- Software Unit
- Non-Unit DB
- TO DB

Program code: Accessing relations

You can get relations, its relation type and the name of the related object in several ways.

Modify the following program code for getting the relation composition:

```
private void AccessRelationComposition(PlcUnitProvider provider)
{
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelationComposition unitRelations = m_PlcUnit.Relations;
}
```

Modify the following program code for getting relation by indexer:

```
private void AccessRelationByIndex(PlcUnitProvider provider)
{
m_PlcUnit = provider.UnitGroup.Units[0]; //assuming existing units
PlcUnitRelation unitRelation = m_PlcUnit.Relations[1];
}
```

Modify the following program code for getting relations by iterating through them:

```
private void AccessUnitRelations(PlcUnitProvider  provider)
{
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelationComposition unitRelations = m_PlcUnit.Relations;
foreach (PlcUnitRelation relation in unitRelations)
{
// using 'relation'
}
}
```

Modify the following program code for getting the relation type of a unit by the RelationType attribute:

```
private void AccessUnitRelationType(PlcUnitProvider  provider)
{
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
UnitRelationType unitRelationType = m_PlcUnit.Relations[2].RelationType;
}
```

Modify the following program code for getting the name of the related object of a unit by the RelatedObject attribute:

```
private void AccessUnitRelatedObject(PlcUnitProvider provider)
{
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
string unitRelatedObjectName = m_PlcUnit.Relations[1].RelatedObject;
}
```

Modify the following program code to identify relation in the collection using Find with the name of accessible element (RelatedObject):

```
private void SearchUnitRelationFromRelatedObject(PlcUnitProvider provider)
{
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelation relation = m_PlcUnit.Relations.Find(unitRelatedObjectName);
}
```

5.11.6.5 Updating software unit properties

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to update unit properties such as Author and Comment. You cannot set and get the comment property directly as it is a MultilingualText. The Comment Item's text property can be updated.

The Comment.Item contains the cultures of the TIA Portal project. It can be adjusted via value assignment. The project languages are indexed in the Comment.Item composition, 0 marked as the first default project language. If there are more languages to be set, you can iterate through it, otherwise if the culture does not exist, an EngineeringTargetInvocationException is thrown.

Program code: Update unit's author property

Modify the following program code to modify unit's author property.

```
private static void UpdateUnitProperty(PlcUnit unit1, PlcUnit unit2, string  newAuthor =
"Z012345")
{
//Set value
unit1.Author = newAuthor;
//Using SetAttributes()
var attrList = new List<KeyValuePair<string,  object>>()
{
new KeyValuePair<string, object>("Author", newAuthor)
};
unit1.SetAttributes(attrList));
//Using SetAttribute()
IEngineeringObject unit = (IEngineeringObject)unit2;
unit.SetAttribute("Author", newAuthor);
}
```

Error Message

Error will occur if you try to use the wrong value, for example starts with space, or contains illegal characters. An Recoverable Exception will be thrown with the following error message: "The value of attribute 'Author' contains an invalid character at Position 0."

```
private static void UpdateUnitProperty(PlcUnitProvider provider, string plcUnitName)
{
PlcUnitComposition unitComposition = provider.UnitGroup.Units;
//Creating Unit
PlcUnit unit1 = unitComposition.Create(plcUnitName);
try
{
unit1.Author = " Author";
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
```

Program code: Update unit's comment property

You cannot set or get the unit 'Comment' property directly as it is a MultilingualText. You can update the Comment Item's Text Property. The Comment.Item contains the cultures of the TIA Project. It can be adjusted via value assignment. The project languages are indexed in the Comment.Item composition, 0 marked as the first default project language. If there are more languages set, you can iterate on it, otherwise if the culture does not exist, an EngineeringTargetInvocationException is thrown.

```
private static void UpdateUnitProperty(PlcUnit unit1)
{
//Setting the default first culture comment Item text:
unit1.Comment.Items[0].Text = "new Comment";
//Setting other culture comment Item Text:
unit1.Comment.Items[1].Text = "neuro Kommentar";
}
```

Error Message

Error will occur if you try to use the SetAttribute, SetAttributes, GetAttribute, GetAttributes. An EngineeringNotSupportedException will be thrown with the following error message: "Comment is not supported by type 'Siemens.Engineering.SW.Units.PlcUnit'."

```
private static void UpdateUnitProperty(PlcUnit unit1)
{
try
{
  ((IEngineeringObject)unit1).SetAttribute("Comment", "new comments");
}
catch (EngineeringNotSupportedException e)
{
  Console.WriteLine(e.Message);
}
}
```

Error will occur if you try to refer a non existing culture. An EngineeringTargetInvocationException will be thrown with the following error message: "The argument 'index' (3) is outside the valid value range."

```
private static void UpdateUnitProperty(PlcUnit unit1)
{
try
{
 unit1.Comment.Items[3].Text = "new Comment";
}
catch (EngineeringTargetInvocationException e)
{
 Console.WriteLine(e.Message)
}
}
```

5.11.6.6 Publishing software unit object

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness .
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to get and set the Access attribute of the underlying objects of units so that you can control their accessibilty between units.

The following objects are supported in Openness units:

- Program blocks and data blocks - except OBs
- PLC types

The following possible value of the Access attribute represented as enumeration UnitAccessType:

- Published
- Unpublished

Note

The attribute 'Access' is only available on objects located under a unit, so if it is going to be set or get not under PlcUnit related objects a EnginneringNotSupportedException is thrown. In case of a wrong value (e.g.invalid enum values) EngineeringTargetInvocationException is thrown.

Program code: Configuring the Access attribute of PLC blocks

```
private void AccessAttributeOfPLCBlocks(PlcUnit plcUnit, string blockName = "FB_1")
{
//Getting attribute value with GetAttribute
...
PlcBlock block = plcUnit.BlockGroup.Find(blockName);
UnitAccessType currentAccessValue = (UnitAccessType)block.GetAttribute("Access");
...
//Getting attribute value with GetAttributes
...
PlcBlock block = plcUnit.BlockGroup.Block.Find(blockName);
List<string> attrList = new List<string>();
{
"Access"
};
IList<object> getAttributeValues = block.GetAttributes(attrList));
…
//Setting attribute value with SetAttribute
…
PlcBlock block = plcUnit.BlockGroup.Blocks.Find(blockName);
UnitAccessType newAccessValue = UnitAccessType.Published;
block.SetAttribute("Access", newAccessValue);
…
//Setting attribute value with SetAttributes
…
PlcBlock block = plcUnit.BlockGroup.Blocks.Find(blockName);
UnitAccessType newAccessValue = UnitAccessType.Published;
IList attrList = new List<KeyValuePair<string, object>>()
{
new KeyValuePair<string, object>("Access", newAccessValue),
};
block.SetAttributes(attrList);
…
}
```

Program code: Configuring the Access attribute of PLC types

```
private void AccessAttributeOfPLCTypes(PlcUnit plcUnit, string typeName = "UDT_1")
{
//Getting attribute value with GetAttribute
PlcType type = PlcUnit.TypeGroup.Types.Find(typeName);
UnitAccessType currentAccessValue = (UnitAccessType)type.GetAttribute("Access");
//Getting attribute value with GetAttributes
PlcType type = m_PlcUnit.TypeGroup.Types.Find(typeName);
List<string> attrList = new  List<string>()
{
"Access"
};
IList<object> getAttributeValues = type.GetAttributes(attrList));
//Setting attribute value with SetAttribute
PlcType type = m_PlcUnit.TypeGroup.Types.Find(typeName);
UnitAccessType newAccessValue = UnitAccessType.Published;
type.SetAttribute("Access", newAccessValue);
//Setting attribute value with SetAttributes
PlcType type = m_PlcUnit.TypeGroup.Types.Find(typeName);
UnitAccessType newAccessValue = UnitAccessType.Published;
IList attrList = new List<KeyValuePair<string, object>>()
{
new KeyValuePair<string, object>("Access", newAccessValue),
};
type.SetAttributes(attrList));
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.6.7 Adding external sources in units

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to add an external source under a software units. You can add the source files such as .SCL,. DB, UDT under units.

You can perform the following tasks with external source group under software units while using Openness API :

- Adding ExternalSourceGroup under units
- Generating blocks/UDTs from an external source under software units
- Exporting source from blocks/UDTs under units

Program code: Adding an external source group

Modify the following program code to add an external source under a software unit:

```
private void AddExternalSourceGroup(PlcUnitProvider unitProvider, string plcUnitName =
"Unit1")
{
PlcUnit newUnit = unitProvider.UnitGroup.Units.Create(plcUnitName);
PlcExternalSource unitExternalSource =
newUnit.ExternalSourceGroup.ExternalSources.CreateFromFile(externalSourceFilename,
GetFilePath(externalSourceFilename));
}
```

Program code: Generating blocks/UDT from source

Modify the following program code to generate blocks\UDTs with void return types from an external source under a software unit:

```
private void GenerateBlocksFromSource(PlcExternalSource unitExternalSource)
{
unitExternalSource.GenerateBlocksFromSource();
}
```

Modify the following program code to generate blocks / UDT with collection of blocks get generated from an external source under a software unit:

```
private static IList<IEngineeringObject> GenerateBlocksFromSource(PlcExternalSource
unitExternalSource)
{
// Blocks/UDTs gets generated only if there are no compilation errors
IList<IEngineeringObject> generatedBlocksUnderUnitList =
unitExternalSource.GenerateBlocksFromSource(GenerateBlockOption.None);
// Blocks/UDTs will be generated regardless of any compilation errors
IList<IEngineeringObject> generatedBlocksUnderUnitList =
unitExternalSource.GenerateBlocksFromSource(GenerateBlockOption.KeepOnError);
}
```

Program code: Exporting source from blocks / UDTs

Modify the following program code to generate source from blocks under a software unit:

```
private void GenerateSourceFromBlocks(PlcUnit newUnit)
{
// For blocks
PlcBlock unitPlcBlock = newUnit.BlockGroup.Blocks.Find(generatedBlockName);
// Generate source from blocks with dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[]{unitPlcBlock}, new
FileInfo(outputFileGeneratedPath), GenerateOptions.WithDependencies);
// Generate source from blocks without dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[]{unitPlcBlock}, new
FileInfo(outputFileGeneratedPath), GenerateOptions.None);
}
```

Modify the following program code to generate source from UDTs under a software unit:

```
private void GenerateSourceFromUDTs(PlcUnit newUnit)
{
// For UDTs
PlcType unitPlcUdt = newUnit.TypeGroup.Types.Find(generatedUdtName);
//Generate source from UDTs with dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[]{unitPlcUdt}, new
FileInfo(outputFileGeneratedPath), GenerateOptions.WithDependencies);
//Generate source from UDTs without dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[]{unitPlcUdt}, new
FileInfo(outputFileGeneratedPath), GenerateOptions.None);
}
```

Program code: Enumerating external source file group

Modify the following program code to enumerate through the external source file group composition under the current unit:

```
private void AccessExternalSourceFileGroup(PlcUnitProvider unitProvider)
PlcUnit newUnit = unitProvider.UnitGroup.Units.Find(textBoxAddNewUnit.Text);
PlcExternalSourceComposition unitExtSrcComposition =
newUnit.ExternalSourceGroup.ExternalSources;
foreach (PlcExternalSource unitExtSrc in unitExtSrcComposition)
{
unitExtSrc.GenerateBlocksFromSource(GenerateBlockOption.None);
}
```

5.11.6.8 Units as mastercopies

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- A unit is created
  See Working with software unit (Page 595)

Introduction

You can use the TIA Portal Openness to copy units as mastercopies to project library and global library.

You can perform the following possible tasks with unit as mastercopy while using TIA Portal Openness:

- Create unit as mastercopy in project library from software units
- Create unit as mastercopy in global library from software units
- Recreate unit from mastercopy of project library to software units
- Recreate unit from mastercopy of global library to software units

Program code

Modify the following program code to create new unit as mastercopy in project library from software units using MasterCopyComposition:

```
private void IMasterCopySource CreateUnitAsMastercopyInProjectLibrary(PlcUnitProvider
m_UnitProvider, string plcUnitName = "Unit_1")
{
PlcUnitProvider m_UnitProvider = plc.GetService<PlcUnitProvider>();
PlcUnitComposition m_UnitComposition = m_UnitProvider.UnitGroup.Units;
PlcUnit m_SoftwareUnit1 = unitComposition.Create("Unit_1");//Assuming existing units
IMasterCopySource m_UnitAsMasterCopy = (IMasterCopySource)m_SoftwareUnit1;//Assuming that
m_SoftwareUnit1 is present in the PLC
m_ProjectLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy);
}
```

Modify the following program code to create unit as mastercopy in global library from software units:

```
private void IMasterCopySource CreateUnitAsMasterCopyInGlobalLibrary(PlcUnit
m_SoftwareUnit1, TiaPortal m_TiaPortal)
{
// ...
IMasterCopySource m_UnitAsMasterCopy = (IMasterCopySource)m_SoftwareUnit1;//Assuming that
m_SoftwareUnit1 is present in the PLC
m_TiaPortal.GlobalLibraries.Open(libfile, OpenMode.ReadWrite);
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
m_GlobalLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy)
// ...
}
```

Error will occur if you try to create a new unit in read-only global library. A recoverable exception will be thrown with the message "Cannot write to read-only libraries".

```
private void IMasterCopySource CreateUnitAsMasterCopyInGlobalLibrary(PlcUnit
m_SoftwareUnit1, TiaPortal m_TiaPortal)
{
IMasterCopySource m_UnitAsMasterCopy = (IMasterCopySource)m_SoftwareUnit1;//Assuming that
m_SoftwareUnit1 is present in the PLC
m_TiaPortal.GlobalLibraries.Open(libfile, OpenMode.ReadOnly);
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
try
{
m_GlobalLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy);
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
}
```

Modify the following program code to recreate a unit from mastercopy of project library to software units in PNV:

```
private void RecreateUnitFromProjectLibrary(Project project, PlcSoftware software, string
plcUnitName = "Unit_2")
{
//...
ProjectLibrary m_ProjectLibrary = project.ProjectLibrary;
PlcUnitProvider m_UnitProvider = software.GetService<PlcUnitProvider>();
PlcUnitComposition m_UnitComposition = m_UnitProvider.UnitGroup.Units;
// ...
MasterCopy mc_Unit_2 = m_ProjectLibrary.MasterCopyFolder.MasterCopies.Find(plcUnitName);
m_UnitComposition.CreateFrom(mc_Unit_2);//Recreate a Unit from Project Library to
SoftwareUnits folder
}
```

Modify the following program code to recreate unit from mastercopy of global library to software units in PNV.

```
private void RecreateUnitFromGlobalLibrary(TiaPorta m_TiaPortal, string plcUnitName =
"Unit_2")
{
// ...
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
// ...
mc_Unit_2 = m_GlobalLibrary.MasterCopyFolder.MasterCopies.Find("Unit_2");
m_UnitComposition.CreateFrom(mc_Unit_2);//Recreate a unit from Global Library to
SoftwareUnits folder
}
```

5.11.6.9 Updating existing relations & create/delete relations

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to modify the relations of a PLC units so that you can manage the accessibility of objects existing in other PLC units or outside PLC units.

You can perform the following types of modification using TIA Portal Openness:

- Creating new relations
- Deleting existing relations
- Modifying existing relations

Program code: Creating new relations

You can create a new relations by providing the appropriate relation type and the name of the related object.

```
private void CreateNewRelation(PlcSoftware plcTarget)
{
private PlcUnitRelation m_SoftwareUnitRelation;
private PlcUnitRelation m_NonUnitDBRelation;
private PlcUnitRelation m_TODBRelation;
//...
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
m_PlcUnit = plcUnitProvider.UnitGroup.Units[0]; //assuming existing units
}
```

Modify the following program code to create a new relation accessing object in another Plc units:

```
private void CreateRelationForAccessObject(PlcUnit m_PlcUnit, string plcUnitName =
"Unit_2")
{
// ...
//assuming Plc unit "Unit_2" is already existing in the Plc
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create(plcUnitName,
UnitRelationType.SoftwareUnit);
// ...
}
```

Modify the following program code to create a new relation accessing PLC data block outside of Plc units:

```
private void CreateRelationForAcccessDataBlock(PlcUnit m_PlcUnit, string dataBlockName =
"Data_block_1")
{
// ...
//assuming Plc data block "Data_block_1" is already existing in the Plc
m_NonUnitDBRelation = m_PlcUnit.Relations.Create(dataBlockName,
UnitRelationType.NonUnitDB);
// ...
}
```

Modify the following program code to create a new relation accessing a Technological object outside of Plc units:

```
private void CreateRelationForAccessTechnologicalObject(PlcUnit m_PlcUnit, string
technologicalObjectName = "SpeedAxis_1")
{
// ...
 //assuming Technological object "SpeedAxis_1" is already existing in the Plc
m_TODBRelation = m_PlcUnit.Relations.Create(technologicalObjectName,
UnitRelationType.TODB);
// ...
}
```

Possible Error cases for creating a new relation:

Modify the following program code to create new relation accessing non existing related object:

```
private void CreateRelationForAccesNonExistingRelatedObject(PlcUnit m_PlcUnit)
{
//...
 //assuming Plc unit "NonExistingUnit" is not existing in the Plc yet
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("NonExistingUnit", UnitRelationType.
SoftwareUnit);
//...
}
```

Note

In the above program code, the relation is created and no exception is thrown.

Modify the following program code to create a new relation by specifying a name for the related object which is not conform with the TIA naming rules:

```
private void CreateRelationForRelatedObject(PlcUnit m_PlcUnit)
{
//...
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create(" Unit_2",
UnitRelationType.SoftwareUnit);
//...
}
```

Note

In the above program code, the relation is not created and a recoverable exception is thrown due to leading spaces.

Modify the following program code to create a new relation from unit to itself:

```
private void CreateRelationFromUnit(PlcUnit m_PlcUnit)
{
//...
 //assuming m_PlcUnit is assigned to Plc unit "Unit_1"
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_1",
UnitRelationType.SoftwareUnit);
//...
}
```

Modify the following program code to create a new relation by specifying an empty string for the name of the related object:

```
private void CreateRelationForRelatedObject(PlcUnit m_PlcUnit)
{
//...
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create(string.Empty,
UnitRelationType.SoftwareUnit);
//...
}
```

Note

In all of the above program codes for error scenarios, the relation is not created and a recoverable exception is thrown.

Program code: Delete existing relations

You can create a new relations by providing the appropriate relation type and the name of the related object.

```
private void DeleteExistingRelation(PlcUnitProvider plcUnitProvider, string plcUnitName =
"Unit_2", string dataBlockName = "DB_Global", string technologicalName = "Axis_TO")
{
private PlcUnitRelation m_SoftwareUnitRelation;
private PlcUnitRelation m_NonUnitDBRelation;
private PlcUnitRelation m_TODBRelation;
// ...
m_PlcUnit = plcUnitProvider.UnitGroup.Units[0]; //assuming existing units
m_PlcUnit2 = plcUnitProvider.UnitGroup.Units[1]; //assuming existing units
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_2",
UnitRelationType.SoftwareUnit);
m_NonUnitDBRelation = m_PlcUnit.Relations.Create("DB_Global", UnitRelationType.NonUnitDB);
m_TODBRelation = m_PlcUnit.Relations.Create("Axis_TO", UnitRelationType.TODB);
}
```

You can delete the relations in several ways.

Modify the following program code to delete the relation accessed by indexer:

```
private void DeleteUnitRelationByIndex(PlcUnit m_PlcUnit)
{
//...
m_PlcUnit.Relations[0].Delete();
//...
}
```

Modify the following program code to delete the relation directly:

```
private void DeleteUnitRelation(PlcUnitRelation m_SoftwareUnitRelation)
{
// ...
m_SoftwareUnitRelation.Delete();
// ...
}
```

Program code: Modify existing relations

You can modify the relations in several ways:

You can create a new relations by providing the appropriate relation type and the name of the related object.

```
private void ModifyingUnitRelation(PlcUnitProvider plcUnitProvider, string plcUnitName =
"Unit_2")
{
private PlcUnitRelation  m_Relation;
m_PlcUnit = plcUnitProvider.UnitGroup.Units[0];
 //assuming existing units
m_PlcUnit2 = plcUnitProvider.UnitGroup.Units[1];
 //assuming existing units
m_Relation = m_PlcUnit.Relations.Create("Unit_2", UnitRelationType.SoftwareUnit);
// ...
}
```

Modify the following program code to update the name of the related object by assigning a new value for the RelatedObject attribute directly:

```
private void ModifyingUnitRelation(PlcUnitRelation  m_Relation)
{
// ...
m_Relation.RelatedObject = "Unit_3";
// ...
}
```

Modify the following program code to update the name of the related object by assigning a new value for the RelatedObject attribute through SetAttribute:

```
private void ModifyingUnitRelation(PlcUnitRelation  m_Relation)
{
//...
m_Relation.SetAttribute("RelatedObject", "Unit_4");
//...
}
```

Modify the following program code to update the name of the related object by assigning a new value for the RelatedObject attribute through SetAttributes:

```
private void ModifyingUnitRelation(PlcUnitRelation  m_Relation)
{
// ...
IList attrList = new List<KeyValuePair<string, object>>();
{
new KeyValuePair<string, object("RelatedObject", "Unit_4")
};
m_Relation.SetAttributes(attrList);
// ...
}
```

Possible error cases for modifying existing relations

Modify the following program code to update a relation to access a non existing related object:

```
private void ModifyingUnitRelation(PlcUnitRelation  m_Relation)
{
//...
//assuming Plc unit "NonExistingUnit" is not existing in the Plc yet
m_Relation.RelatedObject = "NonExistingUnit";
//...
}
```

Note

In the above program code, the relation is modified and no exception is thrown.

Modify the following program code to update a relation by specifying a name for the related object which is not confirm with the TIA naming rules:

```
private void ModifyingUnitRelation(PlcUnitRelation  m_Relation)
{
//...
m_Relation.RelatedObject = " Unit_2";
//...
}
```

Modify the following program code to update a relation to access itself:

```
private void ModifyingUnitRelation(PlcUnitRelation  m_Relation)
{
//...
//assuming m_Relation is defined under Plc unit "Unit_1"
m_Relation.RelatedObject = "Unit_1";
//...
}
```

Note

In the above program code, the relation is not modified and a recoverable exception is thrown.

Modify the following program code to update a relation in a way which would be a duplicate of another already existing with the same relation type and related object:

```
private void ModifyingUnitRelation(PlcUnitRelation  m_Relation)
{
//...
 //assuming a relation with the same relation type and related object is already existing
m_Relation.RelatedObject = "Unit_2";
//...
}
```

Modify the following program code to update a relation by specifying an empty string for the name of the related object:

```
private void ModifyingUnitRelation(PlcUnitRelation  m_Relation)
{
//...
m_Relation.RelatedObject = string.Empty;
//...
}
```

Note

In all of the above program codes for errors scenarios, the relation is not modified and a recoverable exception is thrown.

5.11.6.10 Accessing namespaces for software units

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to set and get the namespace attribute for software unit.

Property Name	Data Type	Access
NamespacePreset	String	Read/Write

Program code

Modify the following program code to set and get the Namespace attribute for unit:

```
private void AccessNamespacesForUnits(PlcUnitProvider unitProvider,string plcUnitName =
"Unit_1")
{
//To set the Namespace attribute
var unit = unitProvider.UnitGroup.Units.Find(plcUnitName);
unit.NamespacePreset = "Siemens.Simatic";
//To get the Namespace attribute
string unitNamespace = unit.NamespacePreset;
}
```

See also

Opening a project (Page 128)

5.11.6.11 Accessing namespaces for program blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to assign the namespaces attribute for program block under software units via attribute set. The namespace assigned to the program blocks can be exported and imported under units via SimaticML.

Property Name	Data Type	Access
Namespace	String	Read/Write

Program code

Modify the following program code to set and get the Namespace attribute:

```
private void static AccessNamespaceForProgramBlock()
var nameSpace = unitBlock.GetAttribute(name:"Namespace");
unitBlock.SetAttribute(name:"Namespace", value:"Unit.Siemens");
```

Namespace attribute in SimaticML

```
</Sections></Interface>
<IsIECCheckEnabled>false</IsIECCheckEnabled>
<Name>Block_1</Name>
<Namespace>Siemens.Simatic</Namespace>
<Number>1</Number>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
<SetENOAutomatically>false</SetENOAutomatically>
<UDABlockProperties/>
<UDAEnableTagReadback>false</UDAEnableTagReadback>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
<MultilingualTextitem ID="2" CompositionName=":Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.6.12 Accessing namespaces for UDT

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to set and get the namespaces to the UDT via GetAttribute and SetAttribute under software units. The namespace assigned to the UDTs can be exported and imported under units via SimaticML.

Property name	Data Type	Access
Namespace	String	Read/Write

Program code

Modify the following program code to set and get the Namespace attribute:

```
private void AccessNamespaceForUDT()
{
//Namespace can be set and get to the UDTs via the property
unitUDT.Namespace = "Motor";
var namespaceValue = unitUDT.Namespace;
//Namespace can be set and get to the UDTs via the attribute set
var namespaceValue = unitUDT.GetAttribute("Namespace");
}
```

Namespace attribute in SimaticML

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
<Engineering version="V18" />
<DocumentInfo>
<Created>2022-08-03T05:28:40.722228Z</Created>
<ExportSetting>None</ExportSetting>
<InstalledProducts>
<Product>
<DisplayName>Totally Integrated Automation Portal</DisplayName>
<DisplayVersion>V18</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>TIA Portal Version Control Interface</DisplayName>
<DisplayVersion>V18</DisplayVersion>
</OptionPackage>
<Product>
<DisplayName>Feature Cycle 2 TIA Portal</DisplayName>
<DisplayVersion>V18</DisplayVersion></Product>
<Product>
<DisplayName>Feature Cycle 5 TIA Portal</DisplayName>
<DisplayVersion>V18</DisplayVersion>
</Product>
<Product>
<DisplayName>STEP 7 Professional</DisplayName>
<DisplayVersion>V18</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>STEP 7 Safety</DisplayName>
<DisplayVersion>V18</DisplayVersion>
</OptionPackage>
</InstalledProducts>
</DocumentInfo>
<SW.Types.PlcStruct ID="0">
<AttributeList>
<Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v5">
<Section Name="None">
<Member Name="Element_1" Datatype="Bool" />
</Section>
</Sections></Interface>
<Name>UDT_1</Name>
<Namespace>Siemens</Namespace>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="3" CompositionName="Title">
```

```
<ObjectList>
<MultilingualTextItem ID="4" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Types.PlcStruct>
</Document>
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.11.6.13 Generating loadable file for software units

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- The TIA Portal project is open
  SeeOpening a project (Page 128)
- The PLC must be compile clean

Introduction

You can use the TIA Portal Openness to generate a loadable file for software units via the following method: `GenerateLoadable(FileInfo path, IEnumerable<PlcUnit> unit, TargetOption targetOption)`

Note

The generation of a loadable file for software units is supported for PLC1518 or its subtype whose firmware is V2.8 or higher.

Restriction

Loadable files can only be created for such software units, that

- do not contain any system blocks
- do not contain any ProDiag blocks
- do not contain any failsafe blocks

- do not contain OBs of any other type than ProgramCycle
- do not contain block with dynamic binding
- do not contain any technological objects
- do not contain any user datatypes

Parameter and Enum

The GenerateLoadable() accepts the following parameters to generate a loadable file for software units:

Parameter name	Data type	Description	Note
path	FileInfo	Specifies the name and the storage location for the loadable file	Already existing loadable files cannot be replaced and the containing directory cannot be created.
units	System.Collections.Generic.IEnumerable<Siemens.Engineering.SW.Units.PlcUnit>	Specifies the software units that are to be included in the loadable file	The PlcProgramLoader V1.0 supports the download of a single software unit only.
targetOption	Siemens.Engineering.SW.Loader.TargetOption	Specifies whether the loadable file is to be created for a real PLC or for a simulated PLC	If TargetOption.PlcSim is passed, all blocks must have been compiled with simulation support

The TargetOption parameter uses the following enum values to generate a loadable file:

Value	Description
TargetOption.None	The default value, should not be used.
TargetOption.Plc	To generate a loadable file for a real PLC.
TargetOption.PlcSim	To generate a loadable file for a simulated PLC.

Program code

To generate a loadable file for software units modify the following program code:

```
private void GenerateLoadableFileSoftwareUnit(string loadablePath)
{
PlcSoftware plc = GetPlc();
var loadableProviderService = plc.GetService<LoadableProvider>();
// Create a variable with name "file" to access the loadable path
var file = new FileInfo(loadablePath);
// Create a variable with name "unitProvider" to access the unit by calling
// the GetService() with the PlcUnitProvider type.
var unitProvider = plc.GetService<PlcUnitProvider>();
// Create a variable with name "units" for a list of software units.
var units = new List<PlcUnit>
{
// Accessing a specific unit with name "Unit_1"
unitProvider.UnitGroup.Units.Find("Unit_1") ?? throw new InvalidOperationException()
};
// The loadableProviderService generates a loadable file for units by calling
// the GenerateLoadable() with respective parameters.
try
{
loadableProviderService.GenerateLoadable(file, units, TargetOption.PlcSim);
}
// An EngineeringTargetInvocation exception is thrown
// if generation of the loadable file fails.
// The exception message contains hints on
// what caused the loadable file creation to fail.
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine("Generating the loadable file failed.");
Console.WriteLine(e);
}
}
```

Exception

The following possible exception message are listed below:

- It is only possible to create a loadable file from a whole software unit, not from a single block inside of a unit.
- The selected blocks are from different PLCs.
- Destination directory not found: '{0}'.
- It is not possible to create a loadable file from failsafe block: '{0}'.
- Destination file already exists: '{0}'.
- The loadable file is not accessible.
- An internal error occured.
- The PLC has to be compiled.
- Internal serialization of loadable file failed.

- Internal serialization failed, because loadable file was disposed.
- The specified path, file name, or both exceed the system-defined maximum length.
- The PLC '{0}' is not supported.
- The parent PLC of the service and of the target are different.
- Some block(s) cannot be simulated. If a block is a library block, use a library with simulation support. Otherwise, select the option "Support simulation during block compilation" in the project properties and recompile the block.
- It is not possible to create a loadable file from a system block: '{0}'.
- Access to file is denied.
- Software units containing PLC data types are not supported.
- It is not possible to create a loadable file from selected object(s).
- It is not possible to create a loadable file from OB '{0}', because it does not have the event class 'Program cycle'.
- It is not possible to create a loadable file from a Prodiag block: '{0}'.
- It'is not possible to create a loadable file from selected object: '{0}'.
- It Is not possible to create a loadable file from block '{0}' with dynamic binding enabled.

See also

Generating loadable file for blocks outside of software units (Page 523)

5.11.7 Functions for fail-safe unit

5.11.7.1 Generating fail-safe unit

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to add the fail-safe program into software units. The Safety-Software-Unit is a system generated, one-and-only SW-Unit which can be hosted by Fail-safe PLCs only. For information on software unit, see Working with software unit (Page 595)

You can generate the fail-safe program under software unit only if the SafetyInUnit option is enabled via ManagementOfFail-safeInSoftwareUnitsEnvironment API. You can create a new

Fail-safe PLC which supports software units. A SafetyUnit is then created automatically which serves as the one and only host of the F-program in software unit environment.

Method

The TIA Portal Openness provides the following methods which can be used to generate failsafe-PLC with fail-safe program:

Method	Descriptiom
bool GenerationOfDefaultFailsafe-Program()	To retrieve the setting for generating the default fail-safe program.
void GenerationOfDefaultFailsafe-Program(bool)	To enable/disable the the generation of the default fails-safe program when creating a new fail-safe PLC.

The TIA Portal Openness provides the following methods which can be used to genearte failsafe program under SafetyUnit:

Method	Descriptiom
bool ManagementOfFailsafeInSoftwareU-nitsEnvironment()	To retrieve the state of the respective setting.
void ManagementOfFailsafeInSoftwareU-nitsEnvironment (bool)	To enable/disable the hosting of the fail-safe program within a Safety Unit.

Program code

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
using Siemens.Engineering.SW.Units;
namespace GenerateFailSafeUnit
{
internal class Program
{
private static void Main(string[] args)
{
// Generate default fail-safe program under PLC:
GlobalSettings globalSettingsService = myTiaPortal.GetService<GlobalSettings>();
// Get the value
bool isDefaultFailsafeProgram = globalSettingsService.GenerationOfDefaultFailsafeProgram();
// Set the value
globalSettingsService.GenerationOfDefaultFailsafeProgram(true);
// Generate fail-safe program under SafetyUnit environment
GlobalSettings globalSettingsService = myTiaPortal.GetService<GlobalSettings>();
// Get the value
bool isSafetyUnitSupported =
globalSettingsService.ManagementOfFailsafeInSoftwareUnitsEnvironment();
// Set the value
globalSettingsService.ManagementOfFailsafeInSoftwareUnitsEnvironment(true);
}
}
}
```

See also

Opening a project (Page 128)

5.11.7.2 Accessing the SafetyUnit

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access safety unit using the type "PlcSafetyUnitComposition" UnitGroup. Once the PLC has been created, the Safety Unit can neither be created nor deleted.

The Safety Unit is a one-and-only system generated Unit.

Attribute

The TIA Openness supports the following property to access the safety unit created under the SoftwareUnits folder in TIA Portal:

Property name	Datatype	Access
SafetyUnits	PlcSafetyUnitComposition	Read/Write

Program code

Modify the following program code to access the fail-safe unit:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
using Siemens.Engineering.SW.Units;
namespace AccessingSafetyUnit
{
internal class Program
{
private static void Main(string[] args)
{
var plcUnitProvider = plcSoftware.GetService<PlcUnitProvider>();
PlcSafetyUnit safetyUnit = plcUnitProvider.UnitGroup.SafetyUnits.Find("SafetyUnit");
PlcSafetyUnitComposition plcSafetyUnit = plcUnitProvider.UnitGroup.SafetyUnits; //- Create
method is not accessible like
//in standard unit and compile error is thrown
}
}
}
```

See also

Opening a project (Page 128)

5.11.7.3 Creating/deleting SafetyUnit relations

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to create and delete the safety unit relation. The TOs and GlobalDBs can be ceated and deleted in the same way as standard unit relations.

Program code

Modify the following program code to create and delete the relations:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
using Siemens.Engineering.SW.Units;
namespace CreateDeleteRelationshipsForTheSafetyUnit
{
internal class Program
{
private static void Main(string[] args)
{
// Create relation
PlcSoftware plcSoftware = ...;
var plcUnitProvider = plcSoftware.GetService<PlcUnitProvider>();
var safetyUnit = plcUnitProvider.UnitGroup.SafetyUnits.Find("SafetyUnit");
var unitRelation = safetyUnit.Relations.Create("Unit_1", UnitRelationType.SoftwareUnit);
// Delete relation
unitRelation.Delete();
}
}
}
```

See also

Opening a project (Page 128)

5.11.7.4 Creating/Accessing SafetyUnit elements/objects

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to create, delete, access, update, and publish the underlying objects such as blocks, tags, UDT of safety unit. You can work with objects under safety unit likewise in the standard unit except for fail-safe system generated blocks. You can also export and import the blocks under safety unit using TIA Portal Openness.

Program code

Modify the following program code to create and access safety unit tags:

```
PlcUnitProvider plcUnitProvider = sw.GetService<PlcUnitProvider>();
PlcSafetyUnit  safetyUnit = plcUnitProvider.UnitGroup.SafetyUnits.Find("SafetyUnit");
// Create tagtable
var tagTable = safetyUnit.TagTableGroup.TagTables.Create("Tag_Table_1");
// Create tag
var newTag = tagTable.Tags.Create("tag_1");
// Delete tag
newTag.Delete();
// Search tag
var expectedTag = tagTable.Tags.Find("tag_1");
```

Modify the following program code to export and import safety unit blocks:

```
PlcSafetyUnit  safetyUnit = plcUnitProvider.UnitGroup.SafetyUnits.Find("SafetyUnit");
var unitBlock = safetyUnit.BlockGroup.Blocks.Find("Block_2");
// Export the block
unitBlock.Export(new FileInfo(@"D:\Export\SafetyBlock_unit.xml"), ExportOptions.None);
// Import of block
safetyUnit.BlockGroup.Blocks.Import(new
FileInfo( @"C:\Users\z003hzed\Desktop\Block1.xml"), ImportOptions.Override);
```

Program code

Modify the following program code:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
using Siemens.Engineering.SW.Units;
namespace CreatingAndAccessingSafetyUnitElement
{
internal class Program
{
private static void Main(string[] args)
{
// To create and access safety unit tags
PlcUnitProvider plcUnitProvider = sw.GetService<PlcUnitProvider>();
PlcSafetyUnit  safetyUnit = plcUnitProvider.UnitGroup.SafetyUnits.Find("SafetyUnit");
// Create tagtable
var tagTable = safetyUnit.TagTableGroup.TagTables.Create("Tag_Table_1");
// Create tag
var newTag = tagTable.Tags.Create("tag_1");
// Delete tag
newTag.Delete();
// Search tag
var expectedTag = tagTable.Tags.Find("tag_1");
// To export and import safety unit blocks
PlcSafetyUnit  safetyUnit = plcUnitProvider.UnitGroup.SafetyUnits.Find("SafetyUnit");
var unitBlock = safetyUnit.BlockGroup.Blocks.Find("Block_2");
// Export the block
unitBlock.Export(new FileInfo(@"D:\Export\SafetyBlock_unit.xml"), ExportOptions.None);
// Import of block
safetyUnit.BlockGroup.Blocks.Import(new
FileInfo( @"C:\Users\z003hzed\Desktop\Block1.xml"), ImportOptions.Override);
}
}
}
```

See also

Opening a project (Page 128)

5.11.7.5 Exporting/Importing supervisions under SafetyUnit

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to export and import supervision under safety units by accessing supervision provider from safety unit prodiag blocks.

The following restrictions are applied on SafetyUnit:

- Creation, deletion, and copy of the stand-alone SafetyUnit is not allowed via TIA Portal Openness
- Download and upload of SafetyUnit is not allowed via TIA Portal Openness
- Copy of the hosted SafetyUnit into mastercopy of project/global library is not allowed in TIA Portal Openness

Program code

Modify the following program code to export and import supervisions under the safetyunit:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
using Siemens.Engineering.SW.Units;
namespace ExportingImportingSupervisionsUnderSafetyUnit
{
internal class Program
{
private static void Main(string[] args)
{
string filePath = @"E:\Temp\...\fileName.xlsx";
PlcSafetyUnit safetyUnit = ...;
// Export from prodiag supervision provider
var proDiagBlock = safetyUnit.BlockGroup.Blocks.CreateFB("Block_1", true, 1,
ProgrammingLanguage.ProDiag);
var proDiagSupervisionProvider = proDiagBlock.GetService<SupervisionProvider>();
var superVisionResult = proDiagSupervisionProvider.ExportSupervisionsToXlsx(new
FileInfo(filePath));
// Import from prodiag supervision provider
ImportOptions importOptions = ImportOptions.None;
var superVisionResult = proDiagSupervisionProvider.ImportSupervisionsFromXlsx(new
FileInfo(filePath), importOptions);
// Export from global supervision provider.
var globalSupervisionProvider = safetyUnit.GetService<SupervisionProvider>();
var superVisionResult = globalSupervisionProvider.ExportSupervisionsToXlsx(new
FileInfo(filePath));
// Import from global supervision provider
var superVisionResult = globalSupervisionProvider.ImportSupervisionsFromXlsx(new
FileInfo(filePath), importOptions);
// Import supervison settings
var superVisionResult = globalSupervisionProvider.ImportSupervisionSettingsFromXlsx(new
FileInfo(filePath), importOptions);
}
}
}
```

See also

Opening a project (Page 128)

5.11.7.6 Publishing blocks under the SafetyUnit

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to publish all the blocks (FC, FBs, DBs, and IDBs) that are created under the safety unit's program blocks folder. All the system generated blocks in the SafetyUnit are excluded from being publishable and a runtime exception is thrown while accesing.

Program code

Modify the following program code to set the published attribute under safetyunit:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
using Siemens.Engineering.SW.Units;
namespace PublishingBlocksUnderSafetyUnit
{
internal class Program
{
private static void Main(string[] args)
{
PlcSafetyUnit safetyUnit = plcUnitProvider.UnitGroup.SafetyUnits.Find("SafetyUnit");
 var safetyBlock = safetyUnit.BlockGroup.Blocks.Find("Block_2");
//To set the access
safetyBlock.SetAttribute("Access", UnitAccessType.Published);
//To get the access
var access = safetyBlock.GetAttribute("Access");
}
}
}
```

See also

Opening a project (Page 128)

5.12 Functions for accessing the data of an HMI device

5.12.1 Screens

5.12.1.1 Creating user-defined screen folders

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to create a user-defined screen folder:

```
//Creates a screen folder
private static void CreateScreenFolder(HmiTarget hmitarget)
{
    ScreenUserFolder myCreatedFolder =
hmitarget.ScreenFolder.Folders.Create("myScreenFolder");
}
```

5.12.1.2 Deleting a screen from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Note

You cannot delete a permanent area. A permanent area is a system screen that is always present.

Program code

Modify the following program code to delete a screen from a specific folder:

```
public static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    ScreenUserFolder screenUserFolder =
hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = screenUserFolder.Screens;
    Screen screen = screens.Find("myScreenName");
    if (screen != null)
    {
        screen.Delete();
    }
}
```

5.12.1.3 Deleting a screen template from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The project contains an HMI device.

Program code

Modify the following program code to delete a screen template from a specific folder:

```
private static void DeleteScreenTemplateFromFolder(HmiTarget hmiTarget)
{
    string templateName = "MyScreenTemplate";
    ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("myScreenTemplateFolder");
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find(templateName);
    if (template != null)
    {
        template.Delete();
    }
}
```

5.12.1.4 Deleting all screens from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

Note

You cannot delete a permanent area. A permanent area is a system screen that is always present.

Program code

Modify the following program code to delete all screens from a specific folder:

```
private static void DeleteAllScreensFromFolder(HmiTarget hmitarget)
//Deletes all screens from a user folder or a system folder
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("myScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    List<Screen> list = new List<Screen>();
    foreach(Screen screen in screens)
    {
        list.Add(screen);
    }
    foreach (Screen screen in list)
    {
        screen.Delete();
    }
}
```

5.12.2 Cycles

5.12.2.1 Deleting a cycle

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The project contains an HMI device.

Application

You cannot delete standard cycles.

You can identify whether cycles have actually been deleted based on the composition in the object model (composition count) of the respective cycle. It is no longer possible to access these cycles.

Program code

Modify the following program code to delete a cycle from an HMI device:

```
public static void DeleteCycle(HmiTarget hmiTarget)
{
    CycleComposition cycles = hmiTarget.Cycles;
    Cycle cycle = cycles.Find("myCycle");
    cycle.Delete();
}
```

5.12.3 Text lists

5.12.3.1 Deleting a text list

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The project contains an HMI device.

Program code

Modify the following program code to delete a selected text list and all associated list entries from an HMI device:

```
public static void DeleteTextList(HmiTarget hmiTarget)
{
    TextListComposition textLists = hmiTarget.TextLists;
    TextList textList = textLists.Find("myTextList");
    textList.Delete();
}
```

5.12.4 Graphic lists

5.12.4.1 Deleting a graphic list

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The project contains an HMI device.

Program code

Modify the following program code to delete a selected graphic list and all associated list entries from an HMI device:

```
private static void DeleteGraphicList(HmiTarget hmiTarget)
{
    GraphicListComposition graphicLists = hmiTarget.GraphicLists;
    GraphicList graphicList = graphicLists.Find("myGraphicList");
    graphicList.Delete();
}
```

5.12.5 Connections

5.12.5.1 Deleting a connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The project contains an HMI device.

Program code

Modify the following program code to delete a selected communication connection from an HMI device:

```
private static void DeleteConnection(HmiTarget hmiTarget)
{
    ConnectionComposition connections = hmiTarget.Connections;
    Connection connection = connections.Find("HMI_connection_1");
    connection.Delete();
}
```

5.12.6 Tag table

5.12.6.1 Creating user-defined folders for HMI tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to create a user-defined folder for HMI tags:

```
private static void CreateUserfolderForHMITags(HmiTarget hmitarget)
// Creates an HMI tag user folder
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagUserFolder myCreatedFolder = folder.Folders.Create("MySubFolder");
}
```

5.12.6.2 Enumerating tags of an HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to enumerate all tags of an HMI tag table:

```
private static void EnumerateTagsInTagtable(HmiTarget hmitarget)
// //Enumerates all tags of a tag table
{
    TagTable table = hmitarget.TagFolder.TagTables.Find("MyTagtable");
    // Alternatively, you can access the default tag table:
    // TagTable defaulttable = hmitarget.TagFolder.DefaultTagTable;

    TagComposition tagComposition = table.Tags;
    foreach (Tag tag in tagComposition)
    {
        // Add your code here
    }
}
```

5.12.6.3 Deleting an individual tag from an HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to delete a specific tag from an HMI tag table:

```
private static void DeleteATag(HmiTarget hmiTarget)
{
string tagName = "MyTag";
TagTable defaultTagTable = hmiTarget.TagFolder.DefaultTagTable;
TagComposition tags = defaultTagTable.Tags;
Tag tag = tags.Find(tagName);
tag.Delete();
}
```

5.12.6.4 Deleting a tag table from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The project contains an HMI device.

Application

You cannot delete the default tag table

Program code

Modify the following program code:

```
// Delete a tag table from a specific folder
private static void DeleteTagTable(HmiTarget hmiTarget)
{
    string tableName = "myTagTable";
    TagSystemFolder tagSystemFolder = hmiTarget.TagFolder;
    TagTableComposition tagTables = tagSystemFolder.TagTables;
    TagTable tagTable = tagTables.Find(tableName);
    tagTable.Delete();
}
```

5.12.7 VB scripts

5.12.7.1 Creating user-defined script folders

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to create a user-defined script subfolder below a system folder or another user-defined folder:

```
private static void CreateFolderInScriptfolder(HmiTarget hmitarget)
//Creates a script user subfolderVBScriptSystemFolder
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbScriptFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbScriptSubFolder = vbScriptFolder.Folders.Create("mySubfolder");
}
```

5.12.7.2 Deleting a VB script from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The project contains an HMI device.

Program code

Modify the following program code to delete a VB script from a specific folder:

```
//Deletes a vbscript from a script folderVBScriptSystemFolder
private static void DeleteVBScriptFromScriptFolder(HmiTarget hmitarget)
{
    VBScriptUserFolder vbscriptfolder =
hmitarget.VBScriptFolder.Folders.Find("MyScriptFolder");
    var vbScripts = vbscriptfolder.VBScripts;
    if (null != vbScripts)
    {
        var vbScript = vbScripts.Find("MyScript");
        vbScript.Delete();
    }
}
```

5.12.7.3 Deleting a user-defined folder of an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

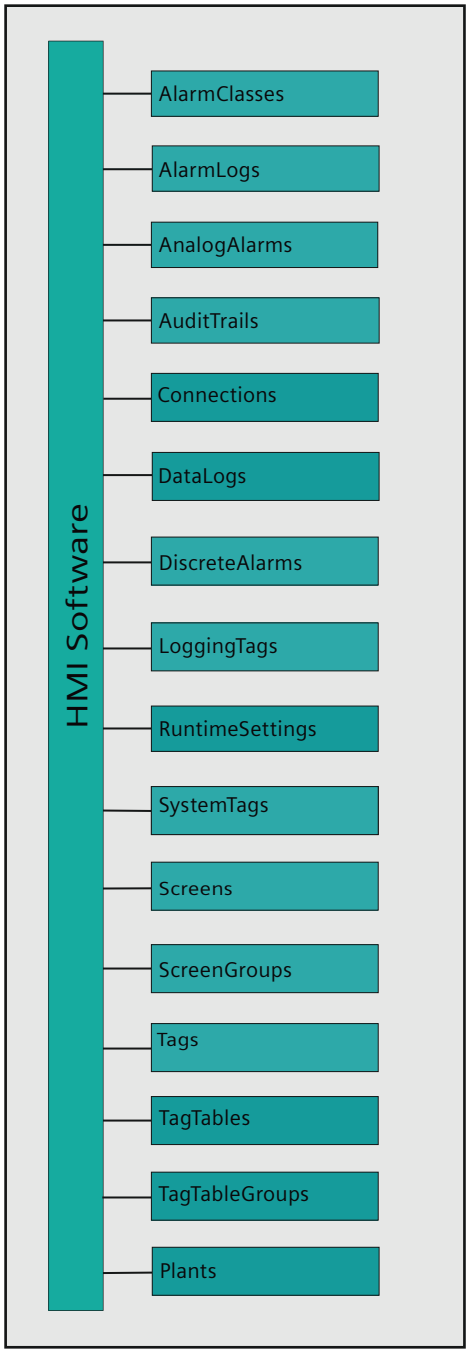
Modify the following program code to delete an user-defined folder of an HMI device:

```
HmiTarget hmiTarget = ...;
ScreenUserFolder screenUserGroup = hmiTarget.ScreenFolder.Folders.Find("MyUserFolder");
screenUserGroup.Delete();
```

5.13 Functions for accessing the data of an HMI Unified device (RT Unified)

5.13.1 HMISoftware (RT Unified)

5.13.1.1 HMISoftware object (RT Unified)


HMI Software
AlarmClasses
AlarmLogs
AnalogAlarms
AuditTrails
Connections
DataLogs
DiscreteAlarms
LoggingTags
RuntimeSettings
SystemTags
Screens
ScreenGroups
Tags
TagTables
TagTableGroups
Plants

5.13.1.2 Description HMISoftware (RT Unified)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  Connecting to the TIA Portal (Page 82)
- A project is open
  Opening a project (Page 128)

Introduction

You can use the HMI Software object to access elements like tags, connections, alarms, and screens.

Program code

To access the HMI Software object modify the following program code:

```
using System;
using System.IO;
using Siemens.Engineering;
using Siemens.Engineering.HmiUnified;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
private HmiSoftware GetHmiSoftware()
{
HmiSoftware hmiSoftware = null;
Project project = null;
IList<TiaPortalProcess> tiaProcessList = TiaPortal.GetProcesses();
//If no TIA Application instance is running, then following using statement will start it or
//If TIA application is already running then will attach to it.
TiaPortal tiaApp = tiaProcessList.Count > 0? tiaProcessList[0].Attach(): new
TiaPortal(TiaPortalMode.WithUserInterface);
// tiaApp.Projects[X] defines the project which has to be opened
// With order number 0 the first project will be opened if several projects are present in
// given TIA instance.
if (tiaApp.Projects.Count > 0)
project = tiaApp.Projects[0];
else
{
//If there is no device project in given TIA instance, then open the existing project.
FileInfo file = new FileInfo(@"D:\Automation\Project1\Project1.apx");
if (!file.Exists)
throw new FileNotFoundException("Project1.apx not found");
project = tiaApp.Projects?.Open(file);
}
//After getting project, get the object representing UA software HMI device.
var devices = project.Devices;
if (devices != null)
{
//If the device is in the first position of the openness project tree
var device = devices[0];
//If the device is not in the first position of the openness project tree, you have to adapt
// the order number devices[X];
var deviceItems = device.DeviceItems;
if (deviceItems != null)
{
foreach (DeviceItem deviceItem in deviceItems)
{
SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
hmiSoftware = softwareContainer?.Software as HmiSoftware;
}
}
}
return hmiSoftware;
}
```

Note

In the above program code, the extension .apx refers to the installed version of TIA Portal.

5.13.1.3 AlarmClasses (RT Unified)

Description AlarmClasses (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on alarm classes you can either use enumerate alarm classes or use Find().

```
private void BrowseAlarmClasses()
{
//User can navigate all alarm classes of device
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
foreach (HmiAlarmClass alarmClass in alarmClasses)
{
//Working with alarm classes
}
}
```

To access a single alarm class by name modify the following program code:

```
private void SearchAlarmClasses()
{
HmiSoftware hmiSoftware = GetHmiSoftware():
// Search for a specific alarm class from list of alarm classes
string alarmClassName = "AlarmClass_1";
HmiAlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
HmiAlarmClass alarmClassObj = alarmsClasses.Find(alarmClassName);
}
```

Working with alarm classes

You can perform the following tasks with tags while using TIA Portal Openness:

- Creating alarm classes:AlarmClasses.Create() (Page 656)
- Deleting alarm classes AlarmClasses.Delete() (Page 656)

- Alarm classes properties: Accessing alarm classes properties (Page 657)
- Alarm classes cross reference service: Accessing cross reference service on alarm classes (Page 659)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

AlarmClasses.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a alarm class with Create() of HmiAlarmClassComposition class modify the following program code:

```
private void AlarmClassCreate()
{
//Create Alarm Class with name "AlarmClass1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
HmiAlarmClass alarmClass = alarmClasses.Create("AlarmClass1");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

AlarmClasses.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a alarm class with Delete() of HmiAlarmClass class modify the following program code :

```
public void DeleteAlarmClass()
//User wants to delete an alarm class with the alarm name AlarmClass1
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;;
HmiAlarmClass alarmClass = alarmClasses.Find("AlarmClass1");
alarmClass.Delete();
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing alarm classes properties (RT Unified)

Property

The following properties are supported in alarm classes:

Property Name	Data Type	Description	Access
Name	String	Specifies the name of an alarm class	Depend on type of alarm class provided (system-based alarm or user-based alarm)
Priority	Byte	Specifies the priority of an alarm class	R/W
IsSystem	Boolean	Specifies if the alarm class is system provided	R
StateMachine	HmiAlarmStateMachine	Specifies the state machine of an alarm class	Depend on type of alarm class provided (system-based alarm or user-based alarm)
RaisedState	HmiRaisedState	Specifies the state for active alarms	R
RaisedState.BackColor	Color	Specifies the back color for active state	R/W
RaisedState.TextColor	Color	Specifies the text color for active state	R/W
RaisedState.Flashing	Boolean	Specifies the flashing for active state	R/W
ClearedState	HmiClearedState	Specifies the state for cleared alarms	R
ClearedState.BackColor	Color	Specifies the back color for cleared state	R/W
ClearedState.TextColor	Color	Specifies the text color for cleared state	R/W
ClearedState.Flashing	Boolean	Specifies the flashing for cleared state	R/W
AcknowledgedState	HmiAcknowledgedState	Specifies the state for acknowledged alarms	R
AcknowledgedState.BackColor	Color	Specifies the back color for acknowledged state	R/W

Property Name	Data Type	Description	Access
AcknowledgedState.TextColor	Color	Specifies the text color for acknowledged state	R/W
AcknowledgedState.Flashing	Boolean	Specifies the flashing for acknowledged state	R/W
AcknowledgedClearedState	HmisAcknowledgedClearedState	Specifies the state for acknowledged and cleared alarms	-
AcknowledgedClearedState.BackColor	Color	Specifies the back color for acknowledged and cleared alarms	R/W
AcknowledgedClearedState.TextColor	Color	Specifies the text color acknowledged and cleared alarms	R/W
AcknowledgedClearedState.Flashing	Boolean	Specifies the flashing for acknowledged and cleared alarms	R/W
Id	UInt32	Specifies the id to identify the alarm class This id is a unique value for each alarm class.	R
Log	System.String	Specifies the log of an alarm class	R/W
CommonAlarmClass	System.String	Specifies the common alarm class	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of alarm classes modify the following program code:

```
private void AlarmClassesPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
HmiAlarmClass alarmClass = hmiSoftware.AlarmClasses.Find("MostCritical");
int priority = alarmClass.Priority;
alarmClass.Priority = 10;
alarmClass.Acknowledgement = StateMachine.AlarmWithDualModeAcknowledgement;
alarmClass.IncomingOutgoingAcknowledgeStatus.BackColor = Color.Red;
alarmClass.IncomingOutgoingAcknowledgeStatus.TextColor = Color.Black;
alarmClass.IncomingOutgoingAcknowledgeStatus.Flashing = true;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing cross reference service on alarm classes (RT Unified)

Requirements

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the cross reference service with GetService() of HmiAlarmClass class modify the following program code:

```
private void getCrossReferenceServiceforTag()
{
SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
HmiSoftware hmiSoftware = softwareContainer.Software as HmiSoftware;
// Accessing lower
// levels until we reach the HmiAlarmClassComposition
HmiAlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
HmiAlarmClass alarmClass = alarmClasses.Find("AlarmClass1");
if (alarmClass != null)
{
try
{
CrossReferenceService crossReferenceService =
alarmClass.GetService<CrossReferenceService>();
......
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
}
}
}
```

See also

HMISoftware object (Page 651)

Opening a project (Page 128)

Connecting to the TIA Portal (Page 82)

5.13.1.4 AlarmLogs (RT Unified)

Description AlarmLogs (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on alarm logs you can either use enumerate alarm logs or use Find().

```
private void AlarmLogBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all the alarm logs of given alarm log table
HmiAlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
foreach (HmiAlarmLog alarmLog in alarmLogs)
{
//working with alarm log.
}
}
```

To access a single alarm log by name modify the following program code:

```
private void SearchAlarmlog()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for a specific alarm log from list of alarmlogs
String alarmLogName = "AlarmLog_1";
HmiAlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
HmiAlarmLog alarmLogObj = alarmLogs.Find(alarmLogName);
}
```

Working with alarm logs

You can perform the following tasks with data logs while using TIA Portal Openness:

- Creating alarm logs: AlarmLogs.Create() (Page 661)
- Deleting alarm logs: AlarmLogs.Delete() (Page 661)
- Alarm logs properties: Accessing alarm logs properties (Page 662)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

AlarmLogs.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a alarm log with Create() of HmiAlarmLogComposition class modify the following program code:

```
private void AlarmLogCreate()
{
//Create Alarmlog
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
HmiAlarmLog alarmLog = alarmLogs.Create("Alarmlog1");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

AlarmLogs.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a alarm log with Delete() of HmiAlarmLog class modify the following program code :

```
private void AlarmLogDelete()
{
//User wants to delete alarm log with name "Alarmlog1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
HmiAlarmLog alarmLog = alarmLogs.Find("Alarmlog1");
alarmLog.Delete();
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing alarm logs properties (RT Unified)

Property

The following properties are supported in the alarm logs:

Property Name		Data Type	Description	Access
Name		String	Specifies the name of alarm log	R/W
Settings		LogSettings	Specifies the members in complex properties such as StorageFolder, LogMaxSize, and LogTimePeriod which define the settings related to alarm log	R
	StoragePath	String	Specifies the path for storage	R/W
	LogMaxSize	Unsigned int	Specifies the maximum size of log on storage medium in units of megabytes	R/W
	LogTimePeriod	LogDuration	Specifies the maximum time period covered by alarm log	R/W
Segment		LogSegment	Specifies the members in complex properties such as SegementMaxSize, StartTime and SegmentTimePeriod which define the settings related to segment	R
	SegmentMaxSize	Unsigned int	Specifies the maximum size of segment of alarm log in units of megabytes	R/W
	SegmentStartTime	DateTime	Specifies the exact point in time when segment is started	R/W
	SegmentTimePeriod	SegmentDuration	Specifies the maximum time period covered by segment of alarm log	R/W
Backup		LogBackup	Specifies the members of complex properties such as PrimaryPath and BackupMode which define the settings related to log backup	R
	BackupMode	HmiBackupMode	Specifies the mode for backup	R/W
	PrimaryPath	String	Specifies the path for backup	R/W

The following properties are supported in log duration:

Property Name	Data Type	Description	Access
Days	Unsigned int	Specifies the set and get number of days	R/W
Hours	Unsigned int	Specifies the set and get number of hours	R/W
Segment	LogSegment	Specifies the members of complex properties such as SegementMaxSize, StartTime and SegmentTimePeriod which define the settings related to Segment	R
Backup	LogBackup	Specifies the members of complex properties such as PrimaryPath and BackupMode which define the settings related to log backup	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of alarm logs modify the following program code:

```
private void AlarmLogPropertiesAccess()
{
//Set and Get alarm log properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
HmiAlarmLog alarmLog = alarmLogs.Find("Alarmlog_1");
string name = alarmLog.Name;
alarmLog.Settings.StorageFolder = @"C:\Logs\Log";
alarmLog.Settings.LogMaxSize = Int32.MaxValue;
alarmLog.Settings.LogMaxSize = 2000;
LogDuration duration = alarmLog.Settings.LogTimePeriod;
double durationInDouble = duration.GetDoubleLogDuration();
string durationInString = duration.GetStringLogDuration();
duration.Days = 7;
duration.Hours = 23;
duration.Minutes = 50;
duration.Seconds = 0;
duration.Ticks = 1;
//Log duration can be set by using function also.
duration.SetLogDuration(10, 12, 4, 5, 0);
alarmLog.Backup.BackupMode = BackupMode.NoBackup;
alarmLog.Backup.BackupMode = BackupMode.PrimaryPath;
alarmLog.Backup.PrimaryPath = @"C:\Logs\Backup";
alarmLog.Segment.SegmentMaxSize = 500;
alarmLog.Segment.SegmentStartTime = DateTime.Now;
alarmLog.Segment.SegmentStartTime = DateTime.Now.Date;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.5 AnalogAlarms (RT Unified)

Description AnalogAlarms (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on analog alarms you can either use enumerate analog alarms or use Find().

```
private void AnalogAlarmBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all analog alarms of device
HmiAnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
foreach (HmiAnalogAlarm analogAlarm in analogAlarms)
{
//Working with analog alarms.
}
}
```

To access a single analog alarm by name modify the following program code:

```
private void SearchAnalogAlarm()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for a specific analog alarm from list of analog alarms
String analogAlarmName = "AnalogAlarm_1";
HmiAnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
HmiAnalogAlarm analogAlarmObj = analogAlarms.Find(analogAlarmName);
}
```

Working with analog alarms

You can perform the following tasks with analog alarms while using TIA Portal Openness:

- Creating tags:AnalogAlarms.Create() (Page 664)
- Deleting tags: AnalogAlarms.Delete() (Page 665)
- Analog Alarm properties: Accessing analog alarm properties (Page 666)
- Analog Alarm cross reference service: Accessing cross reference service on analog alarms (Page 667)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

AnalogAlarms.Create() (RT Unified)

Requirements

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a analog alarm with Create() of HmiAnalogAlarmComposition class modify the following program code:

```
private void AnalogAlarmCreate()
{
//Create an Analog Alarm with name "Alarm1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
HmiAnalogAlarm analogAlarm = analogAlarms.Create("Alarm1");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

AnalogAlarms.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a analog alarm with Delete() of HmiAnalogAlarm class modify the following program code :

```
private void AnalogAlarmDelete()
{
//Delete an Analog Alarm
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
HmiAnalogAlarm analogAlarm = analogAlarms.Find("Alarm_1");
if (analogAlarm != null)
{
analogAlarm.Delete();
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing analog alarm properties (RT Unified)

Property

The following properties are supported in analog alarm:

Property Name	Data Type	Description	Access
EventText	MultilingualText	Specifies the alarm text of an analog alarm	R/W
Id	UInt32	Specifies the id of an alarm. This id is unique value for each analog alarm.	R/W
AlarmClass	String	Specifies the class of an analog alarm.	R/W
Name	String	Specifies the name of an analog alarm.	R/W
Priority	Byte	Specifies the priority of an analog alarm.	R/W
Origin	String	Specifies the origin of an analog alarm	R/W
Area	String	Specifies the area of an analog alarm	R/W
RaisedState-Tag	String	Specifies the tag that triggers the analog alarm	R/W
Condition-Value	Object supported type: Double, Int16, UInt16, Int32, UInt32, Int64, UInt64, Byte	Specifies the limit value for analog alarm which can be specified as constant value	R/W
Condition	HmiAlarmCondition	Specifies the limit mode for an analog alarm	R/W
InfoText	MultilingualText	Specifies the info text of an analog alarm	R/W
AlarmPara-meterTags	Object Supported-Type: IList<string>	An array of in maximum 10 elements specifies the alarm parameters in an alarm text. Each element of the array contains a tag name as string.	R/W
EventText1	MultilingualText	Specifies the additional text 1 of an analog alarm	R/W
EventText2	MultilingualText	Specifies the additional text 2 of an analog alarm	R/W
EventText3	MultilingualText	Specifies the additional text 3 of an analog alarm	R/W
EventText4	MultilingualText	Specifies the additional text 4 of an analog alarm	R/W
EventText5	MultilingualText	Specifies the additional text 5 of an analog alarm	R/W
EventText6	MultilingualText	Specifies the additional text 6 of an analog alarm	R/W
EventText7	MultilingualText	Specifies the additional text 7 of an analog alarm	R/W
EventText8	MultilingualText	Specifies the additional text 8 of an analog alarm	R/W
EventText9	MultilingualText	Specifies the additional text 9 of an analog alarm	R/W
TriggerAd-dress	String	Specifies the address of the trigger tag	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of analog alarm modify the following program code:

```
private void AnalogAlarmPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
HmiAnalogAlarm analogAlarm = analogAlarms.Find("Alarm_1");
uint id = analogAlarm.Id;
analogAlarm.Id = 10;
int priority = analogAlarm.Priority;
analogAlarm.Priority = 7;
string area = analogAlarm.Area;
analogAlarm.Area = "Area_1";
string name = analogAlarm.Name;
analogAlarm.Name = "NewAlarm";
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing cross reference service on analog alarms (RT Unified)

Requirements

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the cross reference service with GetService() of HmiAnalogAlarm class modify the following program code:

```
private void getCrossReferenceServiceforAnalogAlarm()
{
SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
HmiSoftware hmiSoftware = softwareContainer.Software as HmiSoftware;
// Accessing lower levels
// until we reach the HmiAnalogAlarmComposition.
HmiAnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
HmiAnalogAlarm analogAlarm = analogAlarms.Find("Alarm1")
if (analogAlarm != null)
{
try
{
CrossReferenceService crossReferenceService =
analogAlarm.GetService<CrossReferenceService>();
......
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.6 AuditTrails (RT Unified)

Description AuditTrails (RT Unified)

Requirements

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on audit trails you can either use enumerate alarm trail or use Find().

```
private void AuditTrailBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all the audit trails
HmiAuditTrailComposition auditLogs = hmiSoftware.AuditTrails
foreach (HmiAuditTrail auditLog in auditLogs)
{
//working with audit trails.
}
}
```

To access a single audit trail by name modify the following program code:

```
private void SearchAuditTrails()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for specific audit trail name
string auditTrailName = "AuditTrail_1";
HmiAuditTrailComposition auditLogs = hmiSoftware.AuditTrails;
HmiAuditTrail auditLogObject = auditLogs [01];
AuditTrail auditTrailObj = auditLogObject.Find(auditTrailName);
}
```

Working with audit trails

You can perform the following tasks with audit trails while using TIA Portal Openness:

- Audit trail properties: Accessing audit trails properties (Page 670)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing audit trails properties (RT Unified)

Property

The following properties are supported in the Audit Trails:

Property Name		Data Type	Description	Access
Name		String	Specifies the name of audit trail	R/W
Settings		LogSettings	Specifies the members of complex properties such as StorageFolder, LogMaxSize, and LogTimePeriod which defines the settings related to the audit trail	R
	StorageFolder	String	Specifies the folder path for storage	R/W
	LogMaxSize	Unsigned int	Specifies the maximum size of log on storage medium in units of megabytes	R/W
	LogTimePeriod	LogDuration	Specificies the maximum time period covered by log	R/W
	StorageDevice	DeviceNode	Specifies the storage device	R/W
Segment		LogSegment	Specifies the members of complex properties such as SegementMaxSize, StartTime, and SegmentTimePeriod which define the settings related to the segment	R
	SegmentMaxSize	Unsigned int	Specifies the maximum size of segment in audit trail in units of megabytes	R/W
	SegmentStartTime	DateTime	Specifies the exact point in time when segment is started	R/W
	SegmentTimePeriod	SegmentDuration	Specifies the maximum time period covered by segment of audit trail	R/W
Backup		LogBackup	Specific the members of complex properties such as PrimaryPath, and BackupMode which define the settings related to the audit trail backup	R
	BackupMode	HmiBackupMode	Specifies the mode for backup	R/W
	PrimaryPath	String	Specifies the primary path for backup	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of audit trails modify the following program code:

```
private void AuditTrailPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiAuditTrailComposition auditLogs = hmiSoftware.AuditTrails;
HmiAuditTrail auditLogObject = auditLogs [01];
string auditTrailName = auditLogObject.Name;
Console.WriteLine("Name {0}" auditTrailName);
auditLogObject.Name = "Audit Log_100";
auditLogObject.Settings.StorageFolder = @"C:\\Testsl23";
auditLogObject.Settings.LogMaxSize = Int32.MaxValue;
auditLogObject.Settings.LogMaxSize = 2000;
LogDuration duration = auditLogObject.Settings.LogTimePeriod;
double durationlnDouble = duration.GetDoubleLogDuration();
string durationlnString = duration .GetStringLogDuration();
duration.Days = 200;
duration.Hours = 10;
duration.Minutes = 22;
duration.Seconds = 39;
duration.Ticks = 0;
//Log duration can be set by function also
duration.SetLogDuration(10,12,4,5,0);
auditLogObject.Backup.BackupMode = HmiBackupMode.NoBackup;
auditLogObject.Backup.BackupMode = HmiBackupMode.PrimaryPath;
auditLogObject.Backup.PrimaryPath = @"C:\Logs\Backup";
auditLogObject.Segment.SegmentMaxSize = 500;
auditLogObject.Segment.SegmentStartTime = DateTime.Now;
auditLogObject.Segment.SegmentStartTime = DateTime.Now.Date;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.7 Connections (RT Unified)

Description Connections (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform task on connection you can either use enumerate connections or use Find( )

```
private void ConnectionBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all connections of device
HmiConnectionComposition connections = hmiSoftware.Connections;
foreach (HmiConnection connection in connections)
{
//working with connection
}
}
```

To access a single connection by name modify the following program code:

```
private void ConnectionSearch()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for specific connection name from a list of connections
string connectionName = "Connection_1";
HmiConnectionComposition connections = hmiSoftware.Connections;
HmiConnection connectionObj = connections.Find(connectionName);
}
```

Working with connections

You can perform the following tasks with connections while using TIA Portal Openness:

- Creating connections: Connections.Create() (Page 672)
- Deleting connections: Connections.Delete() (Page 673)
- Connection properties: Accessing connection properties (Page 674)
- Connection cross reference service: Accessing cross reference service on connections (Page 675)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Connections.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a connection with Create() of HmiConnectionComposition class modify the following program:

```
private void ConnectionCreate()
{
//Create a connection with name "Connection_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.Connections;
HmiConnection connection = connections.Create("Connection_1");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Connections.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a connection with Delete() of HmiConnection class modify the following program code:

```
private void ConnectionDelete()
{
//Delete a connection with name "Connection_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.Connections;
HmiConnection connection = connections.Find("Connection_1");
connection.Delete();
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing connection properties (RT Unified)

Property

The following properties are supported in connection:

Property name	Data type	Description	Access
Name	String	Specifies the name of connection	R/W
DisabledAtStartup	Boolean	Specifies whether the connection is disabled at the start of Runtime	R/W
CommunicationDriver	String	Specifies the communication driver	R/W
Node	String	Specifies the access point of partner (eg PLC).	R
Partner	String	Specifies the name of connected PLC.	R
Station	String	Specifies the name of the station to which PLC is located.	R
Comment	String	Specifies the additional comments if any	R/W
InitialAddress	String	Specifies the parameters of connection like type of interface (DP, TCP/IP), IP address, Rack etc.	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of connection modify the following program code:

```
private void ConnectionPropertiesAccess()
{
// Accessing properties of Connection object.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.Connections;
HmiConnection connection = connections.Find("Connection_12");
string name = connection.Name;
connection.Name = "Connection_15";
bool online = connection.DisabledAtStartup;
connection.DisabledAtStartup = true;
string node = connection.Node;
string station = connection.Station;
string partner = connection.Partner;
string communicationDriver = connection.CommunicationDriver;
connection.CommunicationDriver = "SIMATIC S7 300/400";
// Accessing "InitialAddress" property of connection object
// with valid strings
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet;
HostAddress = 127.157.8.1; HostAccessPoint = S7ONLINE; PlcRack = 5; PlcIsCyclicOperation =
true";
// Accessing "InitialAddress" property of connection object with invalid value or invalid
// format will return recoverable exception.
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing cross reference service on connections (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the cross reference service with GetService() of HmiConnection class modify the following program code:

```
private void getCrossReferenceServiceforConnection()
{
SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
HmiSoftware hmiSoftware = softwareContainer.Software as HmiSoftware;  //Accessing lower
levels
// until we reach the HmiConnectionComposition.
HmiConnectionComposition connections = hmiSoftware.Connections;
HmiConnection hmiConnection = connections.Find("Connection1");
if (hmiConnection != null)
{
try
{
CrossReferenceService crossReferenceService =
hmiConnection.GetService<CrossReferenceService>();
......
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.8 DataLogs (RT Unified)

Description Datalogs (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on data logs you can either use enumerate datalog or use Find().

```
private void DatalogBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all data logs of device
HmiDataLogComposition dataLogs = hmiSoftware.DataLogs;
foreach (HmiDataLog dataLog in dataLogs)
{
// Working with data logs
}
}
```

To access a single data log by name modify the following program code:

```
private void SearchDatalog()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for a specific data log from list of data logs
String datalogName = "DataLog_1";
HmiDataLogComposition dataLogs = hmiSoftware.DataLogs;
HmiDataLog dataLogObj = dataLogs.Find(dataLogName);
}
```

Working with data logs

You can perform the following tasks with data logs while using TIA Portal Openness:

- Creating data logs:Datalogs.Create() (Page 677)
- Deleting data logs: Datalogs.Delete() (Page 678)
- Data logs properties: Accessing data logs properties (Page 679)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Datalogs.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a datalog with Create() of HmiDatalogComposition class modify the following program code:

```
private void DatalogCreate()
{
//Create Datalog
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiDataLogComposition dataLogs = hmiSoftware.DataLogs;
HmiDataLog dataLog = dataLogs.Create("Datalog_1");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Datalogs.Delete() (RT Unified)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a data log with Delete() of HmiDataLog class modify the following program code :

```
private void DatalogDelete()
{
//user wants to delete datalog with name "Datalog1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiDataLogComposition dataLogs = hmiSoftware.DataLogs;
HmiDataLog dataLog = dataLogs.Find("Datalog1");
dataLog.Delete();
}
```

Accessing data logs properties (RT Unified)

Property

The following properties are supported in the data logs:

Property Name		Data Type	Description	Access
Name		String	Specifies the name of data log	R/W
Settings		LogSettings	Specifies the members of complex properties such as StorageFolder, LogMaxSize and LogTimePeriod which define the settings related to log	R
	StorageFolder	String	Specifies the folder path for storage	R/W
	LogMaxSize	Unsigned int	Specifies the maximum size of log on storage medium in units of megabytes	R/W
	LogTimePeriod	LogDuration	Specifies the maximum time period covered by log	R/W
	StorageDevice	DeviceNode	Specifies the storage device to be checked	R/W
Segment		LogSegment	Specifies the members of complex properties such as SegementMaxSize, StartTime and SegmentTimePeriod) which define the settings related to segment	R
	SegmentMaxSize	Unsigned int	Specifies the maximum size of segment of log in units of megabytes	R/W
	SegmentStartTime	DateTime	Specifies the exact point in time when segment of log is started (to be checked)	R/W
	SegmentTimePeriod	SegmentDuration	Specifies the maximum time period covered by segment of log	R/W
Backup		LogBackup	Specifies the members of complex properties such as PrimaryPath and BackupMode which define the settings related to log backup	R
	BackupMode	HmiBackupMode	Specifies the mode for back up	R/W
	PrimaryPath	String	Specifies the primary path for back up	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of data logs modify the following program code:

```
private void DatalogPropertiesAccess()
{
//Set/Get Data Log properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiDataLogComposition dataLogs = hmiSoftware.DataLogs;
HmiDataLog dataLog = dataLogs.Find("Datalog_1");
string name = dataLog.Name;
dataLog.Settings.StorageFolder = @"C:\Logs\Log";
dataLog.Settings.LogMaxSize = Int32.MaxValue;
dataLog.Settings.LogMaxSize = 2000;
LogDuration duration = dataLog.Settings.LogTimePeriod;
double durationInDouble = duration.GetDoubleLogDuration();
string durationInString = duration.GetStringLogDuration();
duration.Days = 7;
duration.Hours = 23;
duration.Minutes = 50;
duration.Seconds = 0;
duration.Ticks = 1;
//Method to set log duration.
duration.SetLogDuration(10, 12, 4, 5, 0);
dataLog.Backup.BackupMode = BackupMode.NoBackup;
dataLog.Backup.BackupMode = BackupMode.PrimaryPath;
dataLog.Backup.PrimaryPath = @"C:\Logs\Backup";
dataLog.Segment.SegmentMaxSize = 500;
dataLog.Segment.SegmentStartTime = DateTime.Now;
dataLog.Segment.SegmentStartTime = DateTime.Now.Date;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.9 DiscreteAlarms (RT Unified)

Description DiscreteAlarms (RT Unified)

Requirements

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on discrete alarms you can either use enumerate discrete alarms or use Find().

```
private void DiscreteAlarmBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all discrete alarms of device
HmiDiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
foreach (HmiDiscreteAlarm discreteAlarm in discreteAlarms)
{
//working with discrete alarms.
}
}
```

To access a single discrete alarm by name modify the following program code:

```
private void SearchDiscreteAlarm()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for a specific discrete alarm from list of discrete alarms
String discreteAlarmName = "DiscreteAlarm_1";
HmiDiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
HmiDiscreteAlarm discreteAlarmObj = discreteAlarms.Find(discreteAlarmName);
}
```

Working with discrete alarms

You can perform the following tasks with tags while using TIA Portal Openness:

- Creating discrete alarm:DiscreteAlarms.Create() (Page 681)
- Deleting discrete alarm: DiscreteAlarms.Delete() (Page 682)
- Discrete Alarm properties: Accessing discrete alarm properties (Page 683)
- Discrete Alarm cross reference service: Accessing cross reference service on discrete alarms (Page 685)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

DiscreteAlarms.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a discrete alarm with Create() of HmiDiscreteAlarmComposition class modify the following program code:

```
private void CreateDiscreteAlarm()
{
//Create a discrete alarm with name "Alarm1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiDiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
HmiDiscreteAlarm discreteAlarm = discreteAlarms.Create("Alarm1");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

DiscreteAlarms.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a discrete alarm with Delete() of HmiDiscreteAlarm class modify the following program code :

```
public void DeleteDiscreteAlarm()
//User wants to delete a discrete alarm with the alarm name Alarm_1
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HDiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
HmiDiscreteAlarm discreteAlarm = discreteAlarms.Find("Alarm1");
discreteAlarm.Delete();
}
```

See also

Opening a project (Page 128)

Connecting to the TIA Portal (Page 82)

Accessing discrete alarm properties (RT Unified)

Property

The following properties are supported in the Discrete Alarm:

Property Name	Data Type	Description	Access
EventText	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
Id	UInt32	Specifies the id of alarm This id is unique value for each discrete alarm.	R/W
AlarmClass	String	Specifies the class of discrete alarm	R/W
Name	String	Specifies the name of discrete alarm	R/W
Priority	Byte	Specifies the priority for discrete alarm	R/W
Origin	String	Specifies the origin of discrete alarm	R/W
Area	String	Specifies the area of discrete alarm	R/W
RaisedStateTag	String	Specifies the tag that triggers the discrete alarm	R/W
RaisedStateTagBitNumber	UInt32	Specifies the trigger bit on trigger tag	R/W
TriggerMode	HmiDiscreteAlarmTiggerMode	Specifies the trigger mode for discrete alarm	R/W
InfoText	MultilingualText	Specifies the info text of discrete alarm	R/W
AlarmParameterTags	Object	If array of 10 parameter tags where each tag specifies alarm parameter. Each alarm parameter takes tag name as string. These parameters can be used in alarm text.	R/W
EventText1	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
EventText2	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
EventText3	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
EventText4	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
EventText5	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
EventText6	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
EventText7	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W

Property Name	Data Type	Description	Access
EventText8	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
EventText9	MultilingualText	Specifies the event text of HmiDiscreteAlarm	R/W
AcknowledgmentStateTag	String	Specifies the acknowledgment tag state for discrete alarm	R/W
AcknowledgmentStateTagBitNumber	UInt32	Specifies the acknowledgment tag bit number for discrete alarm	R/W
AcknowledgmentControlTag	String	Specifies the Control Tag for acknowledgment of discrete alarm	R/W
AcknowledgmentControlTagBitNumber	UInt32	Specifies the control tag bit number for acknowledgment of discrete alarm	R/W
TriggerBitAddress	String	Specifies the address of the trigger bit	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of discrete alarm modify the following program code:

```
private void DiscreteAlarmPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiDiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
HmiDiscreteAlarm discreteAlarm = discreteAlarms.Find(Alarm_1);
uint id = discreteAlarm.Id;
discreteAlarm.Id = 10;
int priority = discreteAlarm.Priority;
discreteAlarm.Priority = 7;
discreteAlarm.area = "Area1";
string area = discreteAlarm.Area;
discreteAlarm.Name = "NewAlarm";
string name = discreteAlarm.Name;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing cross reference service on discrete alarms (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the cross reference service with GetService() of HmiDiscreteAlarm class modify the following program code:

```
private void getCrossReferenceServiceforDiscreteAlarm()
{
SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
HmiSoftware hmiSoftware = softwareContainer.Software as HmiSoftware;  //Accessing lower
levels until we reach the HmiTagComposition.
HmiDiscreteAlarmCompositionAlarmComposition discreteAlarms = hmiSoftware.DisceteAlarms;
HmiDiscreteAlarm discreteAlarm = discreteAlarms.Find("Alarm1");
if (discreteAlarm != null)
{
try
{
CrossReferenceService crossReferenceService =
discreteAlarm.GetService<CrossReferenceService>();
......
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.10 LoggingTags (RT Unified)

Description LoggingTags (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on logging tag you can either use enumerate logging tags or use Find().

```
private void LoggingTagBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all logging tags of device
String hmiTagName = "HMI_Tag_1";
HmiTagComposition hmiTags = hmiSoftware.Tags;
HmiTag hmiTag = hmiTags.Find(hmiTagName);
HmiLoggingTagComposition loggingTags = hmiTag.LoggingTags;
foreach (HmiLoggingTag loggingTag in loggingTags)
{
//Working with logging tags
}
}
```

To access a single logging tag by name modify the following program code:

```
private void SearchLoggingTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for specific logging tag
String hmiTagName = "HMI_Tag_1";
String loggingTagName = "Logging_Tag1";
HmiTagComposition hmiTags = hmiSoftware.Tags;
HmiTag hmiTag = hmiTags.Find(hmiTagName);
HmiLoggingTagComposition loggingTags = hmiTag.LoggingTags";
HmiLoggingTag loggingTagObj = loggingTags.Find(loggingTagName);
}
```

Working with logging tags

You can perform the following tasks with data logs while using TIA Portal Openness:

- Creating logging tags: LoggingTags.Create() (Page 687)
- Deleting logging tags: LoggingTags.Delete() (Page 687)
- Logging tags properties: Accessing logging tags properties (Page 688)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

LoggingTags.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a logging tag for a tag with Create() of HmiLoggingTagComposition class modify the following program code:

```
private void LoggingTagCreate()
{
//Create Logging Tag for given tag.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition hmiTags = hmiSoftware.Tags;
//Tag1 exists in TIA project.
HmiTag hmiTag = hmiTags.Find("Tag1");
HmiLoggingTagComposition loggingTags = hmiTag.LoggingTags;
HmiLoggingTag loggingTag = loggingTags.Create("LoggingTag1");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

LoggingTags.Delete() (RT Unified)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a logging tag for a tag with Delete() of HmiLoggingTag class modify the following program code :

```
private void LoggingTagDelete()
{
//To delete a Logging tag with name "LoggingTag_1" for tag "Tag1";
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition hmiTags = hmiSoftware.Tags;
//Tag1 exists in TIA project.
HmiTag hmiTag = hmiTags.Find("Tag1");
HmiLoggingTagComposition loggingTags = hmiTag.LoggingTags;
//LoggingTag1 exists in TIA project.
HmiLoggingTag loggingTag = loggingTags.Find("LoggingTag1");
loggingTag.Delete();
}
```

Accessing logging tags properties (RT Unified)

Property

The following properties are supported in the Logging Tags:

Property Name	Data Type	Description	Access
AggregationDelay	System.TimeSpan	Specifies the value of aggregation delay	R/W
AggregationMode	HmiAggregationMode	Specifies the value of aggregation mode	R/W
Cycle	System.String	Specifies the logging cycle of logging tag	R/W
CycleFactor	System.UInt32	Specifies the value of logging cycle factor	R/W
DataLog	System.String	Specifies the data log of logging tag	R/W
HighLimit	System.Object	Specifies the high limit of logging tag	R/W
LimitScope	HmiLimitScope	Specifies the limit scope of logging tag	R/W
LoggingMode	HmiLoggingMode	Specifies the logging mode of logging tag	R/W
LowLimit	System. Object	Specifies the low limit of logging tag	R/W
Name	System.string	Specifies the name of logging tag	R/W
SmoothingDeltaValue	System.Double	Specifies the delta value of logging tag	R/W
SmoothingMaxTime	System. TimeSpan	Specifies the maximum time of logging tag	R/W
SmoothingMinTime	System.TimeSpan	Specifies the minimum time of logging tag	R/W

Property Name	Data Type	Description	Access
SmoothingMode	HmiSmoothingMode	Specifies the smoothing mode of logging tag	R/W
Source	System.string	Specifies the source of logging tag	R/W
TriggerMode	HmiTriggerMode	Specifies the trigger mode of logging tag	R/W
TriggerTag	System.String	Specifies the value of trigger tag	R/W
TriggerTagBitNumber	System.UInt32	Specifies the value of trigger tag bit number	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of logging tags modify the following program code:

```
private void LoggingTagPropertiesAccess()
{
//Set and Get LoggingTag properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition hmiTags = hmiSoftware.Tags;
//Tag_1 exists in TIA project.
HmiTag hmiTag = hmiTags.Find("Tag_1");
HmiLoggingTagComposition loggingTags = hmiTag.LoggingTags;
//LoggingTag_1 exists in TIA project.
HmiLoggingTag loggingTag = loggingTags.Find("LoggingTag_1");
string name = loggingTag.Name;
HmiSmoothingMode smoothingMode = loggingTag.SmoothingMode;
loggingTag.SmoothingMode = HmiSmoothingMode.SwingingDoor;
HmiLimitScope limitScope = loggingTag.LimitScope;
loggingTag.LimitScope = HmiLimitScope.WithinLimits;
object lowLimit = loggingTag.LowLimit;
loggingTag.LowLimit = "-32768";
object highLimit = loggingTag.HighLimit;
loggingTag.HighLimit = "-3";
TimeSpan smoothingMinTime = loggingTag.SmoothingMinTime;
TimeSpan tsMin = TimeSpan.Parse("5:00:00");
loggingTag.SmoothingMinTime = tsMin;
TimeSpan smoothingMaxTime = loggingTag.SmoothingMaxTime;
TimeSpan tsMax = TimeSpan.Parse("1000");
loggingTag.SmoothingMaxTime = tsMax;
double smoothingDeltaValue = loggingTag.SmoothingDeltaValue;
loggingTag.SmoothingDeltaValue = 5;
string dataLog = loggingTag.Name;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.11 RuntimeSettings (RT Unified)

Description RuntimeSettings (RT Unified)

Requirement

- The HMI Software object is accessible.
  See Description HMISoftware (Page 653)

Program code

To perform task on runtime settings you can either use enumerate runtime settings or use Find( ):

```
private void RuntimeSettingsBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all runtime settings of device
HmiRuntimeSetting runtimeSettings = hmiSoftware.RuntimeSettings;
foreach (RuntimeSetting runtimeSetting in runtimeSettings)
{
//Working with runtime setting
}
}
```

Working with Runtime settings

You can perform the following tasks with runtime settings while using TIA Portal Openness:

- Runtime setting properties: Accessing runtime settings properties (Page 691)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing runtime settings properties (RT Unified)

Property

The following properties are supported in Runtime Settings:

Property name	Data type	Description	Access
HmiSoftware.RuntimeSettings	HmiRuntimeSetting	Specifies the runtime setting object	R
RuntimeSettings.StartScreen	String	Specifies the start screen.	R/W
RuntimeSettings.OperateAsOpcServer	Boolean	Specifies that given device should act as OPC server.	R/W
RuntimeSettings.LanguageAndFonts	HmiLanguageAndFont Association	Specifies the list of language and fonts	R
RuntimeSettings. LanguageAndFonts [index].Enable	Boolean	Specifies the enable / disable runtime language.	R/W
RuntimeSettings.LanguageAndFonts [index].EnableForLogging	Boolean	Specifies the language use for logging on runtime	R/W
RuntimeSettings. LanguageAndFonts [index].Order	Short	Specifies the order for transferring fonts to device.	R
RuntimeSettings. LanguageAndFonts [index].Language	String	Specifies the name of language	R
RuntimeSettings. LanguageAndFonts [index].FixedFont1	String	Specifies the predefined font family available for font configuration	R
RuntimeSettings. LanguageAndFonts [index].FixedFont2	String	Specifies the predefined font family available for font configuration	R
RuntimeSettings. LanguageAndFonts [index].FixedFont3	String	Specifies the predefined font family available for font configuration	R
RuntimeSettings.LanguageAndFonts[index].FixedFont4	String	Specifies the predefined font family available for font configuration	R
RuntimeSettings.GMPEnabled	Boolean	Specifies that given device should be enabled for audit logging or not.	R/W
RuntimeSettings.OpcUaServerRuntimeSettings	HmiOpcUaServerRuntimeSettings	Specifies the OPCUA settings fo the device.	R/W
RuntimeSettings.MaxLoginRuntimeSettings.MaxLoginErrors	UInt32	Specifies the maximum number of login attempts before user lock	R/W
RuntimeSettings.MaxLoginRuntimeSettings.EnableLockAfterNumberOfAttempts	Boolean	Enable lock after login errors	R/W
RuntimeSettings.ProcessDiagnosticsRuntimeSettings	HmiProcessDiagnosticsRuntimeSettings	Specifies the Runtime setting of Process diagnostics	R/W
RuntimeSettings.HmiReportingSettings	HmiReportingSettings	Specifies the Runtime Settings of Reporting	R/W
RuntimeSettings.BitSelectionStrategyForResourceLists	Enum(BitNumberEvaluationType)	Specifies the bit selection strategy for text and graphic lists	R/W

Requirement

- The HMI Software object is accessible.
  See Description HMISoftware (Page 653)

Program code

```
private void RuntimeSettingsPropertiesAcccess()
{
//Accessing properties of RuntimeSettings
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiRuntimeSetting runtimeSettings = hmiSoftware.RuntimeSettings;
hmiSoftware.runtimeSettings.OpcUaServerRuntimeSettings.ActAsOPCServer = true;
hmiSoftware.runtimeSettings.StartScreen = "Screen_1";
hmiSoftware.runtimeSettings.LanguageAndFonts[0].Enable = true;
hmiSoftware.runtimeSettings.LanguageAndFonts[0].EnableForLogging = true;
hmiSoftware.runtimeSettings.GMPEnabled = true;
//Accessing Maximal login Runtime Settings property
var maxLoginRuntimeSettings = hmiSoftware.RuntimeSettings.MaxLoginRuntimeSettings;
maxLoginRuntimeSetttings.EnableLockAfterNumberOfAttempts = true;
var attempts = maxLoginRuntimeSetttings.EnableLockAfterNumberOfAttempts;
maxLoginRuntimeSettings.MaxLoginErrors = 1;
var maxLoginErrors = maxLoginRuntimeSettings.MaxLoginErrors;
//Accessing Process diagnostics Runtime Settings property
HmiProcessDiagnosticsRuntimeSettings hmiProcessDiagnosticsRuntimeSettings =
hmiSoftware.RuntimeSettings.ProcessDiagnosticsRuntimeSettings;
hmiProcessDiagnosticsRuntimeSettings.EnableProcessDiagnostics = true;
var enableProcessDiagnostics =
hmiProcessDiagnosticsRuntimeSettings.EnableProcessDiagnostics;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.12 SystemTags (RT Unified)

Description SystemTags (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access a single system tag by name modify the following program code:

```
private void AccessSystemTagName()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for specific system tag name from device
string systemTagName = "@SystemHealthIndex"
HmiSystemTagComposition hmiSystemTags = hmiSoftware.SystemTags;
HmiSystemTag hmiSystemTagObj = hmiSystemTags.Find(systemTagName);
}
```

Working with system tags

You can perform the following tasks with system tags while using TIA Portal Openness:

- System tags properties: Accessing system tag properties (Page 693)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing system tag properties (RT Unified)

Property

The following properties are supported in system tags:

Property name	Data type	Description	Access
Name	String	Specifies the name of system tag	R
DataType	String	Specifies the data type of system tag	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of system tags modify the following program code:

```
private void SystemTagPropertiesAccess()
{
//Get the "Name" and "DataType" properties of system tag
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiSystemTagComposition hmiSystemTags = hmiSoftware.SystemTags;
HmiSystemTag hmiSystemTag = hmiSystemTags.Find("@UserName");
string name = hmiSystemTag.Name;
string dataType = hmiSystemTag.DataType;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.13 Screens (RT Unified)

Description Screens (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on a screen you can use the enumerate, find, index, and Contains() method.

To access all the screens of the device modify the following program code:

```
private void ScreenBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all the screens of device
HmiScreenComposition objScreens = hmiSoftware.Screens;
foreach (HmiScreen ObjHmiScreen in objScreens)
{
//Working with Screens
}
}
```

To access screen from screens list on the basis of name modify the following program code:

```
private void ScreenSearch()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can search specific screen from screenlist
string hmiScreenName = "HMI_Screen_1";
HmiScreenComposition objScreens = hmiSoftware.Screens;
HmiScreen objHmiScreen = objScreens.Find(hmiScreenName);
}
```

To access screen from screens list on basis of index modify the following program code:

```
private void SearchScreenByIndex()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmiSoftware.Screens;
HmiScreen objHmiScreen = objScreens[0];
}
```

To check a particular screen exist in screen item list by using Contains method:

```
private void IsScreenExist()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmiSoftware.Screens;
HmiScreen screen1 = objScreen.Find("TestScreen1");
bool isExist = objScreens.Contains(screen1);
}
```

Working with screens

You can perform the following tasks with screens while using TIA Portal Openness:

- Creating screens:Screens.Create() (Page 696)
- Deleting screens:Screens.Delete() (Page 696)
- Screens properties: Accessing screens properties (Page 697)
- Screens cross reference service: Accessing cross reference service on screens (Page 702)
- Screen event handlers property: EventHandlers (Page 703)
- Screen screenitems property:ScreenItems (Page 705)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Description Screenitems (Page 705)

Description EventHandlers (Page 703)

Screens.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a screen by accessing Screens property of HMISoftware object modify the following program code:

```
private void ScreenCreate()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmiSoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Screens.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a screen by accessing the Screens property of HmiSoftware object modify the following program code:

```
private void DeleteScreen()
{
//Case 1
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmiSoftware.Screens;
HmiScreen objHmiScreen = objScreens.Find("TestScreen");
if (objHmiScreen != null)
{
objHmiScreen.Delete();
}
}
```

```
private void ScreenDelete()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmiSoftware.Screens;
objScreens.Find("TestScreen1");
IEngineeringObject alarmClassEnggObj = (IEngineeringObject)objScreens[0];
if (alarmClassEnggObj != null)
{
alarmClassEnggObj.Invoke("Delete", null);
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing screens properties (RT Unified)

Property

The following properties are supported in screens:

Property Name	Data Type	Description	Access
BackColor	Color	Specifies the background color of screen	R/W
AlternateBackColor	Color	Specifies the screen alternative background color to be used in fill pattern	R/W
Name	String	Specifies the name of screen	R/W
ScreenNumber	UInt16	Specifies the configured number of screen	R/W
BackFillPattern	HmiFillPattern	Specifies the screen background fill pattern	R/W
BackGraphic	String	Specifies graphic to be shown in screen's background.	R/W
HorizontalAlignment	HmiHorizontalAlignment	Specifies the current scrollbar position of the parent Hmi (top level) screen window. If the screen is smaller than its parent window, this alignment is used for positioning	R/W

Property Name	Data Type	Description	Access
VerticalAlignment	HmiVerticalAlignment	Specifies the current scrollbar position of the parent Hmi (top level) screen window. If the screen is smaller than its parent window, this alignment is used for positioning	R/W
BackgroundFillMode	HmiBackgroundFillMode	Specifies if the background fill just screen or entire window view	R/W
BackGraphicStretchMode	HmiGraphicStretchMode	Specifies if the screen background graphic is streched	R/W
DisplayName	String	Specifies the screen display name	R/W
Width	uint	Specifies the dimensions of the screen in device-independend units (DIU)	r/w
Height	uint	Specifies the dimensions of the screen in device-independend units (DIU)	
Enabled (AllowOperatorControl)	Boolean	State whether the screen is operable at all (Enable= "true") or not	R/W
Dynamizations	DynamizationBaseComposition	Specifies the dynamization collection	R
Parent	IEngineeringObject	Specifies the parent of this object	R
ScreenItems	HmiScreenItem BaseComposition	Specifies the screen items composition For more information on screenitems, see ScreenItems (Page 705)	R/W
EventHandlers	HmiScreenEventHandlerComposition	Specifies the event handlers For more information on EventHandlers, see EventHandlers (Page 703)	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To set properties of screen modify the following program code:

```
private void SettingPropertiesScreen()
{
HmiSoftware hmisoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Find("TestScreen");
objHmiScreen.Width = 50;
objHmiScreen.ScreenNumber = 1;
objHmiScreen.BackGraphic = "DownArrow";
objHmiScreen.Enabled = true;
objHmiScreen.Height = 123;
}
```

To get all properties of screen using GetAttributes ( ) modify the following program code:

```
private void GettingPropertiesScreen()
{
HmiSoftware hmisoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Find("TestScreen");
List<string> getlststring = new List<string>();
getlststring.Add("AlternateBackColor");
getlststring.Add("BackColor");
getlststring.Add("BackFillPattern");
getlststring.Add("BackGraphic");
getlststring.Add("BackGraphicStretchMode");
getlststring.Add("BackgroundFillMode");
getlststring.Add("DisplayName");
//getlststring.Add("Dynamization");
getlststring.Add("Enabled");
//getlststring.Add("EnabledExplicitRelease");
getlststring.Add("Height");
getlststring.Add("HorizontalAlignment");
//getlststring.Add("Layers");
getlststring.Add("Name");
//getlststring.Add("Parent");
//getlststring.Add("ScreenElements");
//getlststring.Add("ScreenItems");
getlststring.Add("ScreenNumber");
getlststring.Add("VerticalAlignment");
getlststring.Add("Width");
var getallpropValue = objHmiScreen.GetAttributes(getlststring);
foreach(var item in getallpropValue)
{
Console.Write(item.ToString());
Console.Write("\n");
}
}
```

To set all properties of screen using SetAttributes( ) modify the following program code:

```
private void SettingPropertyValueScreen()
{
UInt32 Height=400;
UInt32 Width = 200;
ushort ScreenNumber = 11;
HmiSoftware hmisoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Find("Screen_1");
Dictionary<string, object> setPropertyName = new Dictionary<string, object>();
setPropertyName.Add("AlternateBackColor", Color.Aqua);
setPropertyName.Add("BackColor", Color.Aqua);
setPropertyName.Add("BackFillPattern", HmiFillPattern.GradientHorizontalTricolor);
setPropertyName.Add("BackGraphic", "DownArrow");
setPropertyName.Add("BackGraphicStretchMode", HmiGraphicStretchMode.Fill);
setPropertyName.Add("BackgroundFillMode", HmiBackgroundFillMode.Screen);
//setPropertyName.Add("DisplayName", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Enabled", false);
//setPropertyName.Add("EnabledExplicitRelease", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Height", Height);
setPropertyName.Add("HorizontalAlignment", HmiHorizontalAlignment.Right);
setPropertyName.Add("Name", "Screen_2");
setPropertyName.Add("ScreenNumber", ScreenNumber);
setPropertyName.Add("VerticalAlignment", HmiVerticalAlignment.Center);
setPropertyName.Add("Width", Width);
objHmiScreen.SetAttributes(setPropertyName);
}
```

To get the value from property using IEngineeringObject's GetAttribute modify the following program code:

```
private void GettingPropertyValueScreen()
{
HmiSoftware hmisoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Find("Testscreen");
IEngineeringObject obj = objHmiScreen;
Color clraltntvbk = (Color)obj.GetAttribute("AlternateBackColor");
Color clrbk = (Color)obj.GetAttribute("BackColor");
HmiFillPattern filptrns = (HmiFillPattern)obj.GetAttribute("BackFillPattern");
string bkgrphic = (string)obj.GetAttribute("BackGraphic");
HmiGraphicStretchMode strmode =
(HmiGraphicStretchMode)obj.GetAttribute("BackGraphicStretchMode");
HmiBackgroundFillMode bkgrndfilmode =
(HmiBackgroundFillMode)obj.GetAttribute("BackgroundFillMode");
MultilingualText dsplyName = (MultilingualText)obj.GetAttribute("DisplayName");
//DynamizationBaseComposition dymztns =
(DynamizationBaseComposition)obj.GetAttribute("Dynamizations");
bool enable = (bool)obj.GetAttribute("Enabled");
//SumRef objsom = (SomRef)obj.GetAttribute("EnableExplicitRelease");
uint Height = (uint)obj.GetAttribute("Height");
HmiHorizontalAlignment hrzntl =
(HmiHorizontalAlignment)obj.GetAttribute("HorizontalAlignment");
HmiVerticalAlignment vrtcle = (HmiVerticalAlignment)obj.GetAttribute("VerticalAlignment");
//HmiLayerPartComposition objlyrs = (HmiLayerPartComposition)Obj.GetAttribute("layers");
string name = (string)obj.GetAttribute("Name");
//IEngineeringObject objparen = (IEngineeringObject)obj.GetAttribute("Parent");
//HmiscreenElementBaseComposition objelemnt =
(HmiscreenElementBaseComposition)obj.GetAttribute("ScreenElements");
//HmiScreenItemBaseComposition objitems =
(HmiScreenItemBaseComposition)obj.GetAttribute("ScreenItems");
ushort scrnmumber = (ushort)obj.GetAttribute("ScreenNumber");
uint width = (uint)obj.GetAttribute("Width");
}
```

To determine if all properties of screen have valid values modify the following program code:

```
private void IsValidPropertyValue()
{
HmiSoftware hmiSoftware = GetHmiSoftware ();
HmiScreen objHmiScreen = hmiSoftware.Screens.Create("HmiScreen_1");
IList<IValidator> objectsToValidate = new List<IValidator>();
foreach (var item in hmiSoftware.Screens)
{
objectsToValidate.Add(item);
}
foreach (IValidator validator in objectsToValidate)
{
IList<HmiValidationResult> errors = validator.Validate();
if (errors != null && errors.Count > 0)
{
foreach (var errornotification in errors)
{
var propName = errornotification.PropertyName;
foreach (var errormasage in errornotification.Errors)
{
Console.WriteLine(errormasage);
}
}
}
}
}
```

Note

If during set of properties the set operation is not able to set the value then a Recoverable exception is thrown

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Description Screenitems (Page 705)

Accessing cross reference service on screens (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the cross reference service with GetService() of screen class modify the following program code:

```
private void AccessCrossReferenceServiceforScreen()
{
  SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
  HmiSoftware hmiSoftware = softwareContainer.Software as HmiSoftware; //Accessing lower
levels until we reach the ScreenComposition.
  ScreenComposition screenComposition = hmiSoftware.Screens;
  Screen screen = screenComposition.Find("Screen_1");
  if (screen != null)
  {
  try
  {
  CrossReferenceService crossReferenceService = screen.GetService<CrossReferenceService>();
  // ......
  }
  catch (Exception e)
  {
  Console.WriteLine(e.Message);
  }
  }
}
```

EventHandlers (RT Unified)

Description EventHandlers (RT Unified)

Requirement

- The HMI Unified Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access event handler for screen items modify the following program code:

```
private void AccessEvent(HmiSoftware hmiSoftware);
{
  var screenComp = ((HmiSoftware)targetSWTag).Screens;
  HmiScreen screenOne = screenComp.Create("Screen2");
  HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
  HmiScreenEventHandler screenHandlerSearched =
screenOne.EventHandlers.Find(HmiScreenEventType.Unloaded);
}
```

Working with event for screen items

You can perform the following tasks with event for screen items while using TIA Portal Openness:

- Creating Event Handlers : EventHandlers.Create() (Page 704)
- Deleting Event Handlers EventHandlers.Delete() (Page 704)
- Event Handlers properties: Accessing EventHandlers properties (Page 705)

EventHandlers.Create() (RT Unified)

Requirement

- The HMI Unified Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create events at the screen item level object using EventHandler navigator modify the following program code:

```
private void CreateEvents(HmiSoftware hmiSoftware)
{
  var screenComp = ((HmiSoftware)targetSWTag).Screens;
  HmiScreen screenOne = screenComp.Create(string screenName = "Screen2");
  HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
}
```

EventHandlers.Delete() (RT Unified)

Requirement

- The HMI Unified Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete events for screen and screen items modify the following program code:

```
private void DeletePropertyEvent(HmiSoftware hmisoftware)
{
  HmiScreen hmiscreen = hmisoftware.Screens.Create(Guid.NewGuid().ToString());
  var hmiscreenPropEventDyn = hmiscreen.EventHandlers.Create("Enabled",
PropertyEventType.Change);
  var enabledevent = hmiscreen.EventHandlers.Find("Enabled", PropertyEventType.Change);
  enabledevent.Delete();
}
```

Accessing EventHandlers properties (RT Unified)

Requirement

- The HMI Unified Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of events for screen and screen items modify the following program code :

```
private void AccessEvent(HmiSoftware hmiSoftware)
{
  var screenComp = ((HmiSoftware)targetSWTag).Screens;
  HmiScreen screenOne = screenComp.Create("Screen2");
  HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
  Console.WriteLine(screenOne.Name + ".EventHandler.EventType - {0}",
screenHandler.EventType);
  Console.WriteLine(screenOne.Name + ".EventHandler.Script.Async - {0}",
screenHandler.Script.Async);
  Console.WriteLine(screenOne.Name + ".EventHandler.Script.GlobalDefinitionAreaScriptCode
- {0}",   screenHandler.Script.GlobalDefinitionAreaScriptCode);
  Console.WriteLine(screenOne.Name + ".EventHandler.Script.ScriptCode - {0}",
screenHandler.Script.ScriptCode);
}
```

ScreenItems (RT Unified)

Description Screenitems (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on screenitems you can use enumerate, find, index commands and the Contains() method.

```
private void ScreenItemsBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
// User can navigate all screen items of screen
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
foreach (var item in screenitems)
{
//Working with screenitems
}
}
```

To access a screenitem from screenitems list by name modify the following program code :

```
private void SearchScreenItem()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
string hmiScreenItemName = "Line_1";
HmiScreenItemBase screenitems = hmiScreen.ScreenItems.Find(hmiScreenItemName);
}
```

To access screenitem from screenitems list by index modify the following program code :

```
private void SearchScreenItemByIndex()
{
HmiScreenItemBase screenitem = hmiScreen.ScreenItems[1];
}
```

To check a particular screenitem exist in screenitem list by using Contains method modify the following program code:

```
private void IsScreenItemExist()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_21");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItems_1");
bool isexists = screenitems.Contains(hmiLine);
}
```

Working with screen items

You can perform the following tasks with screen items while using TIA Portal Openness:

- Creating screen items: ScreenItems.Create() (Page 707)
- Deleting screen items: ScreenItems.Delete() (Page 709)
- Screenitem object list: Screenitem objects (Page 711)
- Screenitems cross reference service: Accessing cross reference service on screenitems (Page 802)
- Screenitems property event handlers: PropertyEventHandlers (Page 800)

See also

Screens.Create() (Page 696)

Screens.Delete() (Page 696)

Accessing line properties (Page 727)

Accessing polyline properties (Page 729)

Accessing polygon properties (Page 731)

Accessing ellipse properties (Page 720)

Accessing ellipse segment properties (Page 723)

Accessing circle segment properties (Page 717)

Accessing elliptical arc properties (Page 721)

Accessing circular arc properties (Page 716)

Accessing circle properties (Page 714)

Accessing rectangle properties (Page 733)

Accessing graphic view properties (Page 725)

ScreenItems.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a Hmiline screen items with Create() of HmiScreenItemBaseComposition class modify the following program code:

```
private void ScreenItemCreate()
{
// Creating a Hmiline screenitem
  HmiSoftware hmiSoftware = GetHmiSoftware();
  HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
  HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
  HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItem_1");
}
```

To create a HmiIOField screenitem with Create() of HmiScreenItemBaseComposition class modify the following program code:

```
private void CreateScreenItem()
{
  // Creating a HmiIOField screenitem
  HmiSoftware hmiSoftware = GetHmiSoftware();
  HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
  HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
  HmiIOField hmiIOField = screenitems.Create<HmiIOField>("ScreenItem_1");
}
```

To create a HmiAlarmControl screenitem with Create() of HmiScreenItemBaseComposition class modify the following program code:

```
private void ControlScreenItemCreate()
{
  // Creating a HmiAlarmControl screenitem
  HmiSoftware hmiSoftware = GetHmiSoftware();
  HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
  HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
  HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItem_12");
}
```

To create a faceplate container with create() of HmiScreenItemBaseComposition class modify the following program code:

```
private void createFaceplateContainer(HmiSoftware hmiSoftware, string screenName =
"screen_1",
string faceplatecontainerName = "Faceplate container_1")
{
  HmiScreen hmiScreen= hmiSoftware.Screens.Find(screenName);
  HmiScreenItemBaseComposition screenItems = hmiSoftware.ScreenItems;
  HmiFaceplateContainer faceplate =
screenItems.Create<HmiFaceplateContainer>(faceplatecontainerName);
}
```

Note

To create other screenitems, you can provide the screenitems type name supported in TIA Portal Openness.

ScreenItems.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete hmiLine screenitems modify the following program code

```
private void ScreenItemsDelete()
{
//Case 1
  HmiSoftware hmiSoftware = GetHmiSoftware();
  HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
  HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
  HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItem_1");
  if (hmiLine != null)
  {
  hmiLine.Delete();
  }
}
```

```
private void ScreenItemDelete()
{
//Case 2
  HmiSoftware hmiSoftware = GetHmiSoftware();
  HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
  HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
  HmiLine hmiLine = (HmiLine)screenitems.Find("ScreenItem_12");
  IEngineeringObject ObjhmiLineEnggObj = hmiLine;
  if (ObjhmiLineEnggObj != null)
  {
  ObjhmiLineEnggObj.Invoke("Delete", null);
  }
}
```

To delete faceplate container screenitem modify the following program code:

```
private void DeleteFaceplateInstance(HmiSoftware hmiSoftware, string
faceplateContainerName = "Faceplatecontainer_1")
{
  HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
  HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
  HmiFaceplateContainer faceplate =
(HmiFaceplateContainer)hmiScreen.ScreenItems.Find(faceplateContainerName);
  If (faceplate! = null)
  {
  faceplate.Delete ();
  }
}
```

Screenitem objects (RT Unified)

Introduction

The following screen items are supported in the TIA Portal Openness:

Namespace (s)	Screen object name	Screen Item name	Type	Property
Siemens.Engineering.HmiUnified.UI.Shapes Siemens.Engineering.HmiUnified.UI.Base	Base objects	Circle	HmiCircle	For details on Circle properties, see Accessing circle properties (Page 714)
		Circular arc	HmiCircularArc	For details on Circular arc properties, see Accessing circular arc properties (Page 716)
		Circle Segment	HmiCircleSegment	For details on Circle segment properties, seeAccessing circle segment properties (Page 717)
		Ellipse	HmiEllipse	For details on ellipse properties, see Accessing ellipse properties (Page 720)
		Elliptical arc	HmiEllipticalArc	For details on Elliptical arc properties, see Accessing elliptical arc properties (Page 721)
		Ellipse Segment	HmiEllipseSegment	For details on ellipse segment properties, see Accessing ellipse segment properties (Page 723)
		Graphic view	HmiGraphicView	For details on Graphic view properties, see Accessing graphic view properties (Page 725)
		Line	HmiLine	For details on line properties, see Accessing line properties (Page 727)
		Polyline	HmiPolyline	For details on polyline properties, see Accessing polyline properties (Page 729)
		Polygon	HmiPolygon	For details on polygon properties, see Accessing polygon properties (Page 731)
		Rectangle	HmiRectangle	For details on Rectangle properties, see Accessing rectangle properties (Page 733)
		Text Box	HmiTextBox	For details on Textbox properties, see Accessing textbox properties (Page 735)

Namespace (s)	Screen object name	Screen Item name	Type	Property
Siemens.Engineering.HmiUnified.ModernUI.Widgets Siemens.Engineering.HmiUnified.ModernUI.Base	Elements	Bar	HmiBar	For details on Bar properties, seeAccessing bar properties (Page 738)
		Button	HmiButton	For details on Button properties, see Accessing button properties (Page 740)
		Checkbox	HmiCheckBoxGroup	For details on Checkbox properties, see Accessing check box properties (Page 743)
		Clock	HmiClock	For details on Clock properties, see Accessing clock properties (Page 745)
		Gauge	HmiGauge	For details on Gauge properties, see Accessing gauge properties (Page 748)
		IO Field	HmiIOField	For details on IO Field properties, see Accessing IO Field properties (Page 750)
		Listbox	HmiListBox	For details on Listbox properties, see Accessing listbox properties (Page 752)
		RadioButton	HmiRadioButtonGroup	For details on RadioButton properties, see Accessing radio button properties (Page 756)
		Slider	HmiSlider	For details on Slider properties, see Accessing slider properties (Page 759)
		Switch	HmiToggleSwitch	For details on Switch properties, see Accessing switch properties (Page 762)
		Symbolic IOField	HmisymbolicIOField	For details on Symbolic IO Field properties, see Accessing symbolic IOField properties (Page 764)
		TouchArea	HmiTouchArea	For details on Touch Area properties, see Accessing touch area properties (Page 766)

Namespace (s)	Screen object name	Screen Item name	Type	Property
Siemens.Engineering.HmiUnified.ModernUI.Controls Siemens.Engineering.HmiUnified.ModernUI.Base	Controls	Alarm Control	HmiAlarmControl	For details on Alarm Control properties, see Accessing alarm control properties (Page 768)
		Function Trend Control	HmiFunctionTrendControl	For details on Function Trend Control properties, see Accessing function trend control properties (Page 770)
		Faceplate Container	HmiFaceplateContainer	For details on Faceplate Container properties, see Faceplate Container (Page 787)
		Media Player	HmiMediaControl	For details on Media Player properties, see Accessing media player properties (Page 772)
		Process Control	HmiProcessControl	For details on Process Control properties, see Accessing process control properties (Page 774)
		Parameter Set Control	HmiDetailedParameterControl	For details on Parameter Set Control properties, see Accessing parameter set control properties (Page 776)
		Screen Window	HmiScreenWindow	For details on Screen Window properties, see Accessing screen window properties (Page 778)
		System Diagnosis Control	HmiSystemDiagnosisControl	For details on System Diagnosis Control properties, see Accessing system diagnosis control properties (Page 780)
		Trend Control	HmiTrendControl	For details on Trend Control properties, see Accessing trend control properties (Page 781)
		Trend Companion	HmiTrendCompanion	For details on Trend Companion properties, see Accessing trend companion properties (Page 783)
		Web Control	HmiWebControl	For details on Web Control properties, see Accessing web control properties (Page 785)
	My controls	Custom Web Control Container	HmiCustomWebControlContainer	For details on Custom Web Control Container properties, see Accessing custom web control container properties (Page 795)
	Dynamic widgets	Custom Widget Container	HmiCustomWidgetContainer	For details on Custom Widget Container properties, see Accessing custom widget container properties (Page 798)

Basic objects (RT Unified)

Accessing circle properties (RT Unified)

Property

The following properties are supported in circle screen items:

Property Name	Property Type	Description	Access
BorderColor	Color	Specifies the border color of circle	R/W
AlternateBorder-Color	Color	Specifies the alternative border color of circle	R/W
BackColor	Color	Specifies the background color of circle	R/W
AlternateBack-Color	Color	Specifies the alternate background color of circle	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of circle	R
RotationCenter-Placement	HmiRotationCenter-Placement	Specifies the pivot point of circle	R/W
BorderWidth	byte	Specifies the border width of circle	R/W
BackFillPattern	HmiFillPattern	Specifies the background fill pattern of circle	R/W
FillLevel	byte	Specifies the fill level of circle	R/W
FillDirection	HmiFillDirection	Specifies the fill direction of circle	R/W
DashType	HmiDashType	Specifies the line type of circle	R/W
ShowFillLevel	bool	Specifies the show fill level of circle	R/W
Radius	uint	Specifies the radius of circle	R/W
CenterX	int	Specifies the X position of circle	R/W
CenterY	int	Specifies the Y position of circle	R/W
RotationAngle	short	Specifies the rotation angle of circle	R/W
RotationCenterX	float	Specifies the X pivot point of circle	R/W
RotationCenterY	float	Specifies the Y pivot point of circle	R/W
Opacity	float	Specifies the opacity of circle The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of circle The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies the visiblity of circle	R/W
TabIndex	ushort	Specifies the tab index of circle	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of circle	R/W
Authorization	string	Specifies the authorization of circle	R/W
EventHandlers	HmiCircleEventHandler-Composition	Specifies the event handle of circle	R
ShowFocusVisu-al	bool	Specifies if the show focus visual is enabled or not	R/W

Property Name	Property Type	Description	Access
Dynamizations	DynamizationBaseComposition	Specifies the dynamization of circle	R
PropertyEventHandlers	PropertyEventHandlerComposition	Specifies the property event handle of circle	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of circle screen items modify the following program code:

```
private void CircleScreenItemsPropertiesAccess
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiCircle circle = screenitems.Create<HmiCircle>("Default Value55647");
//Name
var name = circle.Name;
circle.Name = "Default Value6834e";
//BorderColor
var bordercolor = circle.BorderColor;
circle.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = circle.BorderWidth;
circle.BorderWidth = 10;
//ToolTipText
var tooltip = circle.ToolTipText;
var tooltiptext = circle.ToolTipText.Items[0].Text;
circle.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//DashType
var dashtype = circle.DashType;
circle.DashType = HmiDashType.Solid;
}
```

See also

ScreenItems.Create() (Page 707)

Accessing circular arc properties (RT Unified)

Property

The following properties are supported in circular arc screen items:

Property Name	Property Type	Description	Access
StartAngle	int	Specifies the start angle of circular arc	R/W
AngleRange	int	Specifies the angle range of circular arc	R/W
LineColor	Color	Specifies the line color of circular arc	R/W
AlternateLineColor	Color	Specifies the alternative border color of circular arc	R/W
DashType	HmiDashType	Specifies the line type of circular arc	R/W
EndType	HmiLineEndType	Specifies the end line of circular arc	R/W
StartType	HmiLineEndType	Specifies the start line of circular arc	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of circular arc	R
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot point of circle arc	R/W
CapType	HmiCapType	Specifies the cap type of circular arc	R/W
LineWidth	byte	Specifies the line width of circular arc	R/W
Radius	uint	Specifies the radius of circular arc	R/W
CenterX	int	Specifies the X position of circular arc	R/W
CenterY	int	Specifies the Y position of circular arc	R/W
RotationAngle	short	Specifies the rotation angle of circular arc	R/W
RotationCenterX	float	Specifies the X pivot point of circular arc	R/W
RotationCenterY	float	Specifies the Y pivot point of circular arc	R/W
Opacity	float	Specifies the opacity of circle arc The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of circular arc The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies the visiblity of circular arc	R/W
TabIndex	ushort	Specifies the tab index of circular arc	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of circular arc	R/W
Authorization	string	Specifies the authorization of circular arc	R/W
EventHandlers	HmiCircularArcEventHandlerComposition	Specifies the event handle of circular arc	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of circlular arc screen items modify the following program code:

```
private void CircularArcScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiCircularArc circulararc = screenitems.Create<HmiCircularArc>("Default Value57");
//Name
var name = circulararc.Name;
circulararc.Name = "Default Value567";
//LineColor
var linecolor = circulararc.LineColor;
circulararc.LineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//LineWidth
var linewidth = circulararc.LineWidth;
circulararc.LineWidth = 10;
//ToolTipText
var tooltip = circulararc.ToolTipText;
var tooltiptext = circulararc.ToolTipText.Items[0].Text;
circulararc.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//CapType
var captype = circulararc.CapType;
circulararc.CapType = HmiCapType.Round;
}
```

See also

ScreenItems.Create() (Page 707)

Accessing circle segment properties (RT Unified)

Property

The following properties are supported in circle segment screen items:

Property Name	Property Type	Description	Access
StartAngle	int	Specifies the start angle of circle segment	R/W
AngleRange	int	Specifies the angle range of circle segment	R/W
BorderColor	Color	Specifies the border color of circle segment	R/W

Property Name	Property Type	Description	Access
AlternateBorderColor	Color	Specifies the alternative border color of circle segment	R/W
BackColor	Color	Specifies the background color of circle segment	R/W
AlternateBackColor	Color	Specifies the alternate background color of circle segment	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of circle segment	R
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot point of circle segment	R/W
BorderWidth	byte	Specifies the border width of circle segment	R/W
BackFillPattern	HmiFillPattern	Specifies the background fill pattern of circle segment	R/W
FillLevel	byte	Specifies the fill level of circle segment	R/W
FillDirection	HmiFillDirection	Specifies the fill direction of circle segment	R/W
DashType	HmiDashType	Specifies the line type of circle segment	R/W
ShowFillLevel	bool	Specifies the show level of circle segment	R/W
Radius	uint	Specifies the radius of circle segment	R/W
CenterX	int	Specifies the X position of circle segment	R/W
CenterY	int	Specifies the Y position of circle segment	R/W
RotationCenterX	float	Specifies the X pivot point of circle segment	R/W
RotationCenterY	float	Specifies the Y pivot point of circle segment	R/W
RotationAngle	short	Specifies the rotation angle of circle segment	R/W
Opacity	float	Specifies the opacity of circle segment The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of circle segment The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies the visiblity of circle segment	R/W
TabIndex	ushort	Specifies the tab index of circle segment	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of circle segment	R/W
Authorization	string	Specifies the authorization of circle segment	R/W
EventHandlers	HmiCircleSegmentEventHandlerComposition	Specifies the event handle of circle segment	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Property Name	Property Type	Description	Access
Dynamizations	DynamizationBaseComposition	Specifies the dynamization of circle segment	R
PropertyEventHandlers	PropertyEventHandlerComposition	Specifies the property event handler of circle segment	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of circle segment screen items modify the following program code:

```
private void CircleSegmentScreenItemPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiCircleSegment circlesegment = screenitems.Create<HmiCircleSegment>("Default Value5987");
//Name
var name = circlesegment.Name;
circlesegment.Name = "Default Value59876";
//AlternateBorderColor
var bordercolor = circlesegment.AlternateBorderColor;
circlesegment.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = circlesegment.BorderWidth;
circlesegment.BorderWidth = 10;
//ToolTipText
var tooltip = circlesegment.ToolTipText;
var tooltiptext = circlesegment.ToolTipText.Items[0].Text;
circlesegment.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></
body>";
}
```

See also

ScreenItems.Create() (Page 707)

Accessing ellipse properties (RT Unified)

Property

The following properties are supported in ellipse screen items :

Property name	Property type	Description	Access
BorderColor	Color	Specifies the border color of ellipse	R/W
AlternateBorderColor	Color	Specifies the alternative border color of ellipse	R/W
BackColor	Color	Specifies the background color of ellipse	R/W
AlternateBackColor	Color	Specifies the alternative background color of ellipse	R/W
BorderWidth	byte	Specifies the border width of ellipse	R/W
BackFillPattern	HmiFillPattern	Specifies the background fill pattern of ellipse	R/W
FillLevel	byte	Specifies the fill level of ellipse	R/W
FillDirection	HmiFillDirection	Specifies the fill direction of ellipse	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of ellipse	R
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot point of ellipse	R/W
DashType	HmiDashType	Specifies the line type of ellipse	R/W
ShowFillLevel	bool	Specifies the fill level of ellipse	R/W
RadiusX	uint	Specifies the X radius of ellipse	R/W
RadiusY	uint	Specifies the Y radius of ellipse	R/W
CenterX	int	Specifies the X position of ellipse	R/W
CenterY	int	Specifies the Y position of ellipse	R/W
RotationAngle	short	Specifies the rotation of ellipse	R/W
RotationCenterX	float	Specifies the X pivot point of ellipse	R/W
RotationCenterY	float	Specifies the Y pivot point of ellipse	R/W
Opacity	float	Specifies the opacity of ellipse The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of ellipse The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
TabIndex	ushort	Specifies the tab index of ellipse	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of ellipse	R/W
Authorization	string	Specifies the authorization of ellipse	R/W
EventHandlers	HmiPolyIgonEventHandlerComposition	Specifies the event handle of ellipse	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of ellipse screen items modify the following program code :

```
private void EllipseScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiEllipse ellipse = screenitems.Create<HmiEllipse>("Default Value5");
//Name
var name = ellipse.Name;
ellipse.Name = "Default Value5";
//AlternateBorderColor
var bordercolor = ellipse.AlternateBorderColor;
ellipse.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = ellipse.BorderWidth;
ellipse.BorderWidth = 10;
//ToolTipText
var tooltip = ellipse.ToolTipText;
var tooltiptext = ellipse.ToolTipText.Items[0].Text;
ellipse.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//BackFillPattern
var backfill = ellipse.BackFillPattern;
ellipse.BackFillPattern = HmiFillPattern.GradientBackwardDiagonal;
}
```

See also

ScreenItems.Create() (Page 707)

Accessing elliptical arc properties (RT Unified)

Property

The following properties are supported in elliptical arc screen items:

Property Name	Property Type	Description	Access
StartAngle	int	Specifies the start angle of elliptical arc	R/W
AngleRange	int	Specifies the angle range of elliptical arc	R/W
LineColor	Color	Specifies the line color of elliptical arc	R/W
AlternateLineColor	Color	Specifies the alternative border color of elliptical arc	R/W
StartType	HmiLineStartType	Specifies the start line of elliptical arc	R/W

Property Name	Property Type	Description	Access
EndType	HmiLineEndType	Specifies the end line of elliptical arc	R/W
CapType	HmiCapType	Specifies the cap type of elliptical arc	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of elliptical arc	R
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot point of elliptical arc	R/W
DashType	HmiDashType	Specifies the line type of elliptical arc	R/W
LineWidth	byte	Specifies the line width of elliptical arc	R/W
RadiusX	uint	Specifies the X radius of elliptical arc	R/W
RadiusY	uint	Specifies the Y radius of elliptical arc	R/W
CenterX	int	Specifies the X position of elliptical arc	R/W
CenterY	int	Specifies the Y position of elliptical arc	R/W
RotationAngle	short	Specifies the rotation angle of elliptical arc	R/W
RotationCenterX	float	Specifies the X pivot point of elliptical arc	R/W
RotationCenterY	float	Specifies the Y pivot point of elliptical arc	R/W
Opacity	float	Specifies the opacity of elliptical arc The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of elliptical arc The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies the visiblity of elliptical arc	R/W
TabIndex	ushort	Specifies the tab index of elliptical arc	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of elliptical arc	R/W
Authorization	string	Specifies the authorization of elliptical arc	R/W
EventHandlers	HmiEllipticalArcEventHandlerComposition	Specifies the event handle of elliptical arc	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access basic property of elliptical arc screen items modify the following program code:

```
private void EllipticalArcScreenItemPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiEllipticalArc ellipticalarc = screenitems.Create<HmiEllipticalArc>("Default
Value53987");
//Name
var name = ellipticalarc.Name;
ellipticalarc.Name = "Default Value123124e";
//LineColor
var linecolor = ellipticalarc.LineColor;
ellipticalarc.LineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//LineWidth
var linewidth = ellipticalarc.LineWidth;
ellipticalarc.LineWidth = 10;
//ToolTipText
var tooltip = ellipticalarc.ToolTipText;
var tooltiptext = ellipticalarc.ToolTipText.Items[0].Text;
ellipticalarc.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></
body>";
//CapType
var captype = ellipticalarc.CapType;
ellipticalarc.CapType = HmiCapType.Round;
}
```

See also

ScreenItems.Create() (Page 707)

Accessing ellipse segment properties (RT Unified)

Property

The following properties are supported in ellipse segment segment screen item:

Property name	Property type	Description	Access
StartAngle	int	Specifies the start angle of ellipse segment	R/W
AngleRange	int	Specifies the angle range of ellipse segment	R/W
BorderColor	Color	Specifies the border color of ellipse segment	R/W
AlternateBorderColor	Color	Specifies the alternative border color of ellipse segment	R/W
BackColor	Color	Specifies the background color of ellipse segment	R/W
AlternateBackColor	Color	Specifies the alternate background color of ellipse segment	R/W
BorderWidth	byte	Specifies the border width of ellipse segment	R/W

Property name	Property type	Description	Access
BackFillPattern	byte	Specifies the background fill pattern of ellipse segment	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of ellipse segment	R
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot point of ellipse segment	R/W
FillLevel	byte	Specifies the fill level of ellipse segment	
FillDirection	HmiFillDirection	Specifies the fill direction of ellipse segment	R/W
DashType	HmiDashType	Specifies the line type of ellipse segment	R/W
ShowFillLevel	bool	Specifies the show level of ellipse segment	R/W
RadiusX	uint	Specifies the X radius of ellipse segment	R/W
RadiusY	uint	Specifies the Y radius of ellipse segment	R/W
CenterX	int	Specifies the X position of ellipse segment	R/W
CenterY	int	Specifies the Y position of ellipse segment	R/W
RotationAngle	short	Specifies the rotation angle of ellipse segment	R/W
RotationCenterX	float	Specifies the X pivot point of ellipse segment	R/W
RotationCenterY	float	Specifies the Y pivot point of ellipse segment	R/W
Opacity	float	Specifies the opacity of ellipse segment The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of ellipse segment The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
TabIndex	ushort	Specifies the tab index of ellipse segment	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of ellipse segment	R/W
Authorization	string	Specifies the authorization of ellipse segment	R/W
EventHandlers	HmiEllipseEventHandlerComposition	Specifies the event handle of ellipse segment	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of ellipse segment screen items modify the following program code:

```
private void EllipseSegmentScreenItemPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiEllipseSegment ellipsesegment = screenitems.Create<HmiEllipseSegment>("Default Value5");
//Name
var name = ellipse segmentsegment.Name;
ellipse segmentsegment.Name = "Default Value56";
//AlternateBorderColor
var bordercolor = ellipse segmentsegment.AlternateBorderColor;
ellipse segmentsegment.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = ellipse segmentsegment.Height;
ellipse segmentsegment.BorderWidth = 10;
//ToolTipText
var tooltip = ellipse segmentsegment.ToolTipText;
var tooltiptext = ellipse segmentsegment.ToolTipText.Items[0].Text;
ellipse segmentsegment.ToolTipText.Items[0].Text =
"<body><p>TestforMultilingualProperty</p></body>";
//BackFillPattern
var backfill = ellipse segmentsegment.BackFillPattern;
ellipse segmentsegment.BackFillPattern = HmiFillPattern.GradientBackwardDiagonal;
}
```

Accessing graphic view properties (RT Unified)

Property

The following properties are supported in graphic view screen items:

Property Name	Property Type	Description	Access
GraphicStretch-Mode	HmiGraphicStretchMode	Specifies the scale background graphic	R/W
FillLevel	byte	Specifies the fill level of graphic view	R/W
FillDirection	HmiFillDirection	Specifies the fill direction of graphic view	False
ShowFillLevel	bool	Specifies the show level of graphic view	False
BackFillPattern	HmiFillPattern	Specifies the background fill pattern of graphic view	R/W
AlternateBack-Color	Color	Specifies the alternate background color of graphic view	R/W
Enabled	Bool	Specifies if the allow operation control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of graphic view	R
RotationCenter-Placement	HmiRotationCenterPlace-ment	Specifies the pivot point of graphic view	R/W

Property Name	Property Type	Description	Access
Top	int	Specifies the Y position of graphic view	R/W
Left	int	Specifies the X position of graphic view	R/W
Width	uint	Specifies the width of graphic view	R/W
Height	uint	Specifies the height of graphic view	R/W
Padding	HmiPaddingPart	Specifies the spacing of graphic view	R
RotationAngle	short	Specifies the rotation angle of graphic view	R/W
RotationCenterX	float	Specifies the X pivot point of graphic view	R/W
RotationCenterY	float	Specifies the Y pivot point of graphic view	R/W
Opacity	float	Specifies the opacity of graphic view The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of graphic view The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies the visiblity of graphic view	R/W
Graphic	string	Specifies the graphic of graphic view	R/W
TabIndex	ushort	Specifies the tab index of graphic view	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of graphic view	R/W
Authorization	string	Specifies the authorization of graphic view	R/W
EventHandlers	HmiGraphicViewEventHandlerComposition	Specifies the event handle of graphic view	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of graphic view screenitems modify the following program code:

```
private void GraphicViewScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiGraphicView graphicview = screenitems.Create<HmiGraphicView>("Default Value35487");
//Name
var name = graphicview.Name;
graphicview.Name = "Default Value765";
//AlternateBackColor
var backcolor = graphicview.AlternateBackColor;
graphicview.AlternateBackColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//Height
var height = graphicview.height;
graphicview.Height = 10;
//ToolTipText
var tooltip = graphicview.ToolTipText;
var tooltiptext = graphicview.ToolTipText.Items[0].Text;
graphicview.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//Padding
var padding = graphicview.Padding;
var bottom = padding.Bottom;
padding.Bottom = 50;
var left = padding.Left;
padding.Left = 60;
var right = padding.Right;
padding.Right = 70;
var top = padding.Top;
padding.Top = 50;
}
```

See also

ScreenItems.Create() (Page 707)

Accessing line properties (RT Unified)

Property

The following properties are supported in line screen item:

Property Name	Property Type	Description	Access
X1	int	Specifies the start point of X - axis	R/W
Y1	int	Specifies the start point of Y - axis	R/W
X2	int	Specifies the end point of X - axis	R/W
Y2	int	Specifies the end point of Y - axis	R/W

Property Name	Property Type	Description	Access
Enabled	bool	Specifies if the allow operator control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of line	R
LineColor	Color	Specifies the color of line	R/W
AlternateLineColor	Color	Specifies the alternative color of line	R/W
DashType	HmiDashType	Specifies the line type	R/W
EndType	HmiLineEndType	Specifies the line end	R/W
StartType	HmiLineStartType	Specifies the line start	R/W
CapType	HmiCapType	Specifies the cap type	R/W
LineWidth	byte	Specifies the width of line	R/W
Top	int	Specifies the Y position of line	R/W
Left	int	Specifies the X position of line	R/W
Width	uint	Specifies the width of line	R/W
Height	uint	Specifies the height of line	False
RotationAngle	short	Specifies the rotation of line	R/W
RotationCenterX	float	Specifies the X pivot of line	R/W
RotationCenterY	float	Specifies the Y pivot of line	R/W
RotationCenter-Placement	HmiRotationCenterPlace-ment	Specifies the pivot point of line	R/W
Opacity	float	Specifies the opacity of line The opacity of line should between 0 and 1	R/W
Name	string	Specifies the name of line The length of name character should be-tween 1 - 128	R/W
Visible	bool	Specifies the visibility of line	R/W
TabIndex	ushort	Specifies the tab index of line	R/W
ToolTipText	MultilingualText	Specifies the tool tip of line	R/W
Authorization	string	Specifies the authorization of line	R/W
EventHandlers	HmiLineEventHandler-Composition	Specifies the event handler of line	R
ShowFocusVisual	bool	Specifies if the show focus visual is ena-bled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of line screen items modify the following code:

```
private void LineScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiLine hmiline = (HmiLine)screenitems.Find("Line_1");
//Name
var name = hmiline.Name;
hmiline.Name = "Default Value";
//AlternateLineColor
var linecolor = hmiline.AlternateLineColor;
hmiline.AlternateLineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//Height
var height = hmiline.Height;
hmiline.Height = 100;
//ToolTipText
var tooltip = hmiline.ToolTipText;
var tooltiptext = hmiline.ToolTipText.Items[0].Text;
hmiline.ToolTipText.Items[0].Text = "<body><p>TestforMultilinugualProperty</p></body>";
//DashType
var dashtype = hmiline.DashType;
hmiline.DashType = HmiDashType.DashDotDot;
}
```

See also

ScreenItems.Create() (Page 707)

Accessing polyline properties (RT Unified)

Property

The following properties are supported in polyline screen item:

Property Name	Property Type	Description	Access
LineColor	Color	Specifies the line color of polyline	R/W
Enabled	bool	Specifies if the allow operator control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of polyline	R
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot point of polyline	R/W
AlternateLineColor	Color	Specifies the alternate line color of polyline	R
DashType	HmiDashType	Specifies the line type of polyline	R
EndType	HmiLineEndType	Specifies the end type of polyline	R

Property Name	Property Type	Description	Access
StartType	HmiLineStartType	Specifies the line start of polyline	R
CapType	HmiCapType	Specifies the cap type of polyline	R
LineWidth	byte	Specifies the line width of polyline	R
JoinType	HmiLineJoinType	Specifies the join type in polyline	R
Top	int	Specifies the Y position of polyline	R
Left	int	Specifies the X position of polyline	R
Width	uint	Specifies the width of polyline	R
Height	uint	Specifies the height of polyline	R
RotationAngle	short	Specifies the rotation of polyline	R
RotationCenterX	float	Specifies the X pivot point of polyline	R
RotationCenterY	float	Specifies the Y pivot point of polyline	R
Opacity	float	Specifies the opacity of polyline The value of Opacity should lies between 0 - 1	R
Name	string	Specifies the name of polyline The length of the name character should lies betweenn 1 - 128 characters	R
Visible	bool	Specifies if the visible is enabled or not	R/W
TabIndex	ushort	Specifies the tab index of polyline	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of polyline	R/W
Authorization	string	Specifies the authorization of polyline	R/W
EventHandlers	HmiPolylineEventHandlerComposition	Specifies the event handle of polyline	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of polyline screen items modify the following program code :

```
private void PolylineScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiPolyline polyline = (HmiPolyline)screenitems.Find("Default Value123");
//Name
var name = polyline.Name;
polyline.Name = "Default Value123";
//AlternateLineColor
var linecolor = polyline.AlternateLineColor;
polyline.AlternateLineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//Height
var height = polyline.Height;
polyline.Height = 100;
//ToolTipText
var tooltip = polyline.ToolTipText;
var tooltiptext = polyline.ToolTipText.Items[0].Text;
polyline.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//Joint Type
var dashtype = polyline.JoinType;
polyline.JoinType = HmiLineJoinType.Miter;
//Points
var points = polyline.Points;
var point = points[0];
var x = point.X;
point.X = 10;
var y = point.Y;
point.Y = 10;
var newPoint = points.Create(15, 100);
}
```

See also

ScreenItems.Create() (Page 707)

Accessing polygon properties (RT Unified)

Property

The following properties are supported in polygon screen items:

Property name	Property type	Description	Access
BorderColor	Color	Specifies the border color of polygon	R/W
AlternateBorderColor	Color	Specifies the alternative border color of polygon	R/W
BackColor	Color	Specifies the background color of polygon	R/W

Property name	Property type	Description	Access
AlternateBackColor	Color	Specifies the alternative background color of polygon	R/W
Points	HmiPointComposition	Specifies the point of polygon	R
BorderWidth	byte	Specifies the border width of polygon	R/W
BackFillPattern	HmiFillPattern	Specifies the background fill pattern of polygon	R/W
FillLevel	byte	Specifies the fill level of polygon	R/W
FillDirection	HmiFillDirection	Specifies the fill direction of polygon	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of polygon	R
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot point of polygon	R/W
DashType	HmiDashType	Specifies the line type of polygon	R/W
ShowFillLevel	bool	Specifies the fill level of polygon	R/W
JoinType	HmiLineJoinType	Specifies the join type in polygon	R/W
Top	int	Specifies the Y position of polygon	R/W
Left	int	Specifies the X position of polygon	R/W
Width	uint	Specifies the width of polygon	R/W
Height	uint	Specifies the height of polygon	R/W
RotationAngle	short	Specifies the rotation of polygon	R/W
RotationCenterX	float	Specifies the X pivot point of polygon	R/W
RotationCenterY	float	Specifies the Y pivot point of polygon	R/W
Opacity	float	Specifies the opacity of polylgon The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of polygon The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
TabIndex	ushort	Specifies the tab index of polygon	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of polygon	R/W
Authorization	string	Specifies the authorization of polygon	R/W
EventHandlers	HmiPolylgonEventHandlerComposition	Specifies the event handle of polygon	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of polygon screen items modify the following program code :

```
private void PolygonScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiPolygon polygon = (HmiPolygon)screenitems.Find("Default Value563");
//Name
var name = polygon.Name;
polygon.Name = "Default Value563";
//AlternateBorderColor
var linecolor = polygon.AlternateBorderColor;
polygon.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//Height
var height = polygon.Height;
polygon.Height = 100;
//ToolTipText
var tooltip = polygon.ToolTipText;
var tooltiptext = polygon.ToolTipText.Items[0].Text;
polygon.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//BackFillPattern
var backfill = polygon.BackFillPattern;
polygon.BackFillPattern = HmiFillPattern.GradientBackwardDiagnol;
//Points
var points = polygon.Points;
var point = points[0];
var x = point.X;
point.X = 10;
var y = point.Y;
point.Y = 10;
var newPoint = points.Create(15, 100);
}
```

See also

Screens.Create() (Page 696)

Accessing rectangle properties (RT Unified)

Property

The following properties are supported in rectangle screen items:

Property Name	Property Type	Description	Access
BorderColor	Color	Specifies the border color of rectangle	R/W
AlternateBorderColor	Color	Specifies the alternative border color of rectangle	R/W
BackColor	Color	Specifies the background color of rectangle	R/W

Property Name	Property Type	Description	Access
AlternateBackColor	Color	Specifies the alternative background color of rectangle	R/W
BorderWidth	byte	Specifies the border width of rectangle	R/W
BackFillPattern	HmiFillPattern	Specifies the background fill pattern of rectangle	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	False
CurrentQuality	HmiQuality	Specifies the connection status of rectangle	True
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot point of rectangle	False
Corners	HmiCornersPart	Specifies the corner of rectangle	True
FillLevel	byte	Specifies the fill level of rectangle	False
FillDirection	HmiFillDirection	Specifies the fill direction of rectangle	False
DashType	HmiDashType	Specifies the line type of rectangle	False
ShowFillLevel	bool	Specifies the show level of rectangle	False
Top	int	Specifies the Y position of rectangle	False
Left	int	Specifies the X position of rectangle	False
Width	uint	Specifies the width of rectangle	False
Height	uint	Specifies the height of rectangle	False
RotationAngle	short	Specifies the rotation angle of rectangle	False
RotationCenterX	float	Specifies the X pivot point of rectangle	False
RotationCenterY	float	Specifies the Y pivot point of rectangle	False
Opacity	float	Specifies the opacity of rectangle The value of Opacity should lie between 0 - 1	False
Name	string	Specifies the name of rectangle The length of the name character should lie betweenn 1 - 128 characters	False
Visible	bool	Specifies if the visible is enabled or not	False
TabIndex	ushort	Specifies the tab index of rectangle	False
ToolTipText	MultilingualText	Specifies the tool tip text of rectangle	False
Authorization	string	Specifies the authorization of rectangle	False
EventHandlers	HmiRectangleEventHandlerComposition	Specifies the event handle of rectangle	True
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	False

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of rectangle screen items modify the following program code:

```
private void RectangleSccreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiRectangle rectangle = screenitems.Create<HmiRectangle>("Default Value3587");
//Name
var name = rectangle.Name;
rectangle.Name = "Default Value57";
//BorderColor
var bordercolor = rectangle.BorderColor;
rectangle.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = rectangle.BorderWidth;
rectangle.BorderWidth = 10;
//ToolTipText
var tooltip = rectangle.ToolTipText;
var tooltiptext = rectangle.ToolTipText.Items[0].Text;
rectangle.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//Corners
var corner = rectangle.Corners;
var bottomleftradius = corner.BottomLeftRadius;
corner.BottomLeftRadius = 50;
var bottomrightradius = corner.BottomRightRadius;
corner.BottomRightRadius = 30;
var topleftradius = corner.TopLeftRadius;
corner.TopLeftRadius = 50;
var toprightradius = corner.TopRightRadius;
corner.TopRightRadius = 30;
}
```

See also

ScreenItems.Create() (Page 707)

Accessing textbox properties (RT Unified)

Property

The following properties are supported in textbox screen items:

Property Name	Property Type	Description	Accessibility
ReadOnly	bool	Specifies the read only access of textbox	R/W
TextWrapping	HmiTextWrapping	Specifies the text wrap of textbox	R/W
VerticalTextAlignment	HmiVerticalAlignment	Specifies the vertical text alignment of textbox	R/W

Property Name	Property Type	Description	Accessibility
HorizontalTextAlignment	HmiHorizontalAlignment	Specifies the horizontal text alignment of textbox	R/W
TextTrimming	HmiTextTrimming	Specifies the text trimming of textbox	R/W
ForeColor	Color	Specifies the foreground color of textbox	R/W
BorderColor	Color	Specifies the border color of textbox	R/W
AlternateBorderColor	Color	Specifies the alternate border color of textbox	R/W
Authorization	string	Specifies the authorization of textbox	R/W
BackColor	Color	Specifies the background color of textbox	R/W
BorderWidth	byte	Specifies the border width of textbox	R/W
CurrentQuality	HmiQuality	Specifies the connection status of textbox	R
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
Font	HmiFontPart	Specifies the font of textbox	R/W
Height	uint	Specifies the height of textbox	R/W
Top	int	Specifies the X- position of textbox	R/W
Left	int	Specifies the Y- position of textbox	R/W
Name	string	Specifies the name of textbox	R/W
RotationAngle	short	Specifies the connection status of textbox	R/W
VisualizeQuality	bool	Specifies if the allow operation control is enabled or not	R/W
AlternateBackColor	Color	Specifies the alternate background color of textbox	R/W
RotationCenterX	float	Specifies the X pivot point of textbox	R/W
RotationCenterY	float	Specifies the Y pivot point of textbox	R/W
Opacity	float	Specifies the opacity of textbox The value of Opacity should lie between 0 - 1	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the rotation angle of textbox	R/W
Visible	bool	Specifies the visibilty of textbox	R/W
Width	uint	Specifies the width of textbox	R/W
TabIndex	ushort	Specifies the tab index of textbox	R/W
ToolTipText	string	Specifies the tool tip text of textbox	R/W
EventHandlers	HmiTextBoxEventHandlerComposition	Specifies the event handle of textbox	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
Text	MultilingualText	Specifies the text of textbox	R/W
Padding	HmiPaddingPart	Specifies the padding of textbox	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of textbox modify the following program code:

```
private void TextboxPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiTextBox textbox = screenitems.Create<HmiTextBox>("Default Value334554");
// To access basic property of textbox
//Name
var name = textbox.Name;
texttbox.Name = "Default Value";
//BorderColor
var bordercolor = textbox.BorderColor;
textbox.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = textbox.BorderWidth;
textbox.BorderWidth = 10;
// To access multilingualproperty
//ToolTipText
var tooltip = textbox.ToolTipText;
var tooltiptext = textbox.ToolTipText.Items[0].Text;
textbox.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//To access other typical property of textbox
//Font
var font = textbox.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var.underline = font.Underline;
font.Underline = false;
font.Bold = true;
}
```

Elements (RT Unified)

Accessing bar properties (RT Unified)

Property

The following properties are supported in bar screen items:

Property Name	Property Type	Description	Access
AlternateBackColor	Color	Specifies the alternate background color of bar	R/W
AlternateBorderColor	Color	Specifies the alternate border color of bar	R/W
Authorization	string	Specifies the authorization of bar	R
BackgColor	Color	Specifies the background color of bar	R/W
BorderColor	Color	Specifies the border color of bar	R/W
BarMode	HmiBarMode		R/W
BorderWidth	byte	Specifies the border width of bar	R/W
CurrentQuality	HmiQuality	Specifies the connection status of bar	R
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
Font	HmiFontPart	Specifies the font type of bar	R/W
Label	HmiTextPart	Specifies the label of bar	R/W
Height	uint	Specifies the height of bar	R/W
StraightScale	HmiStraightScalePart	Specifies the linear scale of bar	R/W
Top	int	Specifies the X position of bar	R/W
Left	int	Specifies the Y position of bar	R/W
Name	string	Specifies the name of bar The length of the name character should lie betweenn 1 - 128 characters	R/W
NormalRangeColor	Color	Specifies the normal range color of bar	R/W
OriginValue	double	Specifies the origin value of bar	R/W
OutputFormat	string	Specifies the output format of bar The value of ouput format should lie between 1 - 128	R/W
PeakIndicators	HmiPeakIndicator	Specifies the peak indicator of bar	R/W
Process Value	string	Specifies the process value of bar	R/W
ProcessValueIndicator-BackColor	Color	Specifies the background color process value indicator of bar	R/W
ProcessValueIndicator-ForeColor	Color	Specifies the foreground color process value indicator of bar	R/W
ProcessValueIndicator-Mode	HmiProcessIndicator-Mode	Specifies the mode of process value indicator of bar	R/W

Property Name	Property Type	Description	Access
RelativeToOrigin	bool	Specifies the percentage calculation of orgin value of bar	R/W
RotationAngle	short	Specifies the rotation angle of bar	R/W
VisualizeQuality	bool	Specifies the connection quality of bar	R/W
RotationCenterX	float	Specifies the X pivot point of bar	R/W
RotationCenterY	float	Specifies the Y pivot point of bar	R/W
Opacity	float	Specifies the opacity of bar The value of Opacity should lie between 0 - 1	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of bar	R/W
ScaleBackColor	uint16	Specifies the scale background color of bar	R/W
ScaleForeColor	Color	Specifies the scale foreground color of bar	R/W
ShowTrendIndicator	bool	Specifies the show trend indicator of bar	R/W
Visible	bool	Specifies the visiblity of bar	R/W
Width	uint	Specifies the width of bar	R/W
TabIndex	ushort	Specifies the tab index of bar	R/W
TrendIndicateColor	Color	Specifies the trend indicator color of bar	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of bar	R/W
EventHandlers	HmiBarEventHandlerComposition	Specifies the event handler of bar	R/W
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
Title	HmiTextPart	Specifies the title of bar	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of bar screen items modify the following program code:

```
private void BarScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiBar bar = screenitems.Create<HmiBar>("Default Value334554");
//Name
var name = bar.Name;
bar.Name = "Default Value";
//RotationCenterPlacement
var rotation = bar.RotationCenterPlacement;
bar.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;
//Width
var width = bar.Width;
bar.Width = 100;
//ToolTipText
var tooltip = bar.ToolTipText;
var tooltiptext = bar.ToolTipText.Items[0].Text;
bar.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//Font
var title = bar.Title;
title.Visible = false;
title.ForeColor = Color.Black;
var font = title.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.Name;
font.Name = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Weight = HmiFontWeight.Bold;
}
```

Accessing button properties (RT Unified)

Property

The following properties are supported in button screen items:

Property Name	Property Type	Description	Accessibility
AlternateBackgroundColor	Color	Specifies the alternate background color of button	R/W
AlternateBorderColor	Color	Specifies the alternate border color of button	R/W

Property Name	Property Type	Description	Accessibility
BackgroundColor	Color	Specifies the background color of button	R/W
BorderColor	Color	Specifies the border color of button	R/W
BorderWidth	byte	Specifies the border width of button	R/W
Contents	HmiContentPart	Specifies the content of button	R/W
CurrentQuality	HmiQuality	Specifies the connection status of button	R
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
Font	HmiFontPart	Specifies the font of button	R/W
ForegroundColor	Color	Specifies the foreground color of button	R/W
Graphic	string	Specifies the graphic of button The value of graphic should lie between 1 - 128	R/W
Height	uint	Specifies the height of button	R/W
Top	int	Specifies the Y position of button	R/W
Left	int	Specifies the X position of button	R/W
Rotation	short	Specifies the rotation of button	R/W
VisualizeQuality	bool	Specifies the visualize quality of button	R/W
RotationCenterX	float	Specifies the X pivot point of button	R/W
RotationCenterY	float	Specifies the Y pivot point of button	R/W
Style Item Appearance	HmiStyleItem Appearance	Specifies the style of button	R/W
Opacity	float	Specifies the opacity of button The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of button The length of the name character should lie betweenn 1 - 128 characters	R/W
Padding	HmiPaddingPart	Specifies the padding of button	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of button	R/W
TabIndex	ushort	Specifies the tab index of button	R/W
Text	MultiLingualText	Specifies the text of button	R
ToolTipText	MultilingualText	Specifies the tool tip text of button	R
Visible	bool	Specifies the visiblity of button	R/W
Width	uint	Specifies the width of button	R/W
EventHandlers	buttonEventHandlerComposition	Specifies the event handler of button	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Property Name	Property Type	Description	Accessibility
AlternateGraphic	string	Specifies the graphic with pressed button The value of Alternate Graphic should lie between 1 - 128	R/W
AlternateText	MultiLingualText	Specifiies the text with pressed button	False

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of button screen items modify the following program code:

```
private void ButtonScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiButton button = screenitems.Create<HmiButton>("Default Value354");
//Name
var name = button.Name;
button.Name = "Default Value";
//AlternateBackColor
var altbackcolor = button.AlternateBackColor;
button.AlternateBackColor = Color.Beige;
//Height
var height = button.Height;
button.Height = 100;
//ToolTipText
string culture = "en-US";
Language lang = Tiaproject.LanguageSettings.Languages.Find(new CultureInfo(culture));
MultilingualText mltprop = button.ToolTipText;
MultilingualTextItemComposition textItemComp = mltprop.Items;
MultilingualTextItem mlttextitem = textItemComp.Find(lang);
mlttextitem.Text = <body><p>"TestforMultilingualProperty"</p></body>;
//Padding
var padding = button.Padding;
var bottom = padding.Bottom;
padding.Bottom = 50;
padding.Left = 60;
padding.Right = 70;
padding.Top = 50;
}
```

Accessing check box properties (RT Unified)

Property

The following properties are supported in checkbox screen items:

Property Name	Property Type	Description	Access
AlternateBackground-Color	Color	Specifies the alternate background color of checkbox	R/W
AlternateBorderColor	Color	Specifies the alternate border color of checkbox	R/W
Authorization	string	Specifies the authorization of checkbox	R/W
BackgroundColor	Color	Specifies the background color of checkbox	R/W
BorderColor	Color	Specifies the border color of checkbox	R
BorderWidth	byte	Specifies the border width of checkbox	R/W
Content	HmiContentPart	Specifies the content of checkbox	R/W
CurrentQuality	HmiQuality	Specifies the connection status of checkbox	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
Font	HmiFontPart	Specifies the font of checkbox	R
ForegroundColor	Color	Specifies the foreground color of checkbox	R/W
Height	uint	Specifies the height of checkbox	R/W
Top	int	Specifies the Y position of checkbox	R/W
Left	int	Specifies the X position of checkbox	R/W
Name	string	Specifies the name of checkbox The length of the name character should lie betweenn 1 - 128 characters	R/W
Rotation	short	Specifies the rotation of checkbox	R/W
VisualizeQuality	bool	Specifies the connection quality of checkbox	R/W
RotationCenterX	float	Specifies the X pivot point of checkbox	R/W
RotationCenterY	float	Specifies the Y pivot point of checkbox	R/W
Opacity	float	Specifies the opacity of checkbox The value of Opacity should lie between 0 - 1	R/W
Process value	string	Specifies the process value of checkbox	R/W

Property Name	Property Type	Description	Access
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of checkbox	R/W
SelectionItemHeight	uint16	Specifies the selected height of checkbox	R/W
SelectionItems	HmiSelectionItemPartComposition	Specifies the selected items of checkbox	R
TabIndex	ushort	Specifies the tab index of checkbox	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of checkbox	R
Visible	bool	Specifies the visiblity of checkbox	R/W
Width	uint	Specifies the width of checkbox	R/W
EventHandlers	HmiCheckBoxGroupEventHandler Composition	Specifies the event handler of checkbox	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
Padding	HmiPaddingPart	Specifies the padding of checkbox	R/W
SelectionPosition	HmiHorizontalAlignment	Specifies the selection position of checkbox	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of checkbox screen items modify the following program code :

```
private void CheckboxScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiCheckbox chkbox = screenitems.Create<HmiCheckbox>("Default Value1354");
//Name
var name = chkbox.Name;
chkbox.Name = "DefaultName";
//AlternateBackColor
var altbackcolor = chkbox.AlternateBackColor;
chkbox.AlternateBackColor = Color.Beige;
//Height
var height = chkbox.Height;
chkbox.Height = 100;
//ToolTipText
var tooltip = chkbox.ToolTipText;
var tooltiptext = chkbox.ToolTipText.Items[0].Text;
chkbox.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
//Font
var font = chkbox.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.Name;
font.Name = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Weight = HmiFontWeight.Bold;
}
```

Accessing clock properties (RT Unified)

Property

The following properties are supported in clock screen item:

Property Name	Property Type	Description	Accessibility
AlternativeBackground-Color	Color	Specifies the alternative back-ground color of clock	R/W
AlternativeBorderColor	Color	Specifies the alternative border color of clcok	R/W
BorderColor	Color	Specifies the border color of clock	R/W
Authorization	string	Specifies the authorization of clock	R

Property Name	Property Type	Description	Accessibility
BackgroundColor	Color	Specifies the background color of clock	R/W
BorderWidth	byte	Specifies the border width of clock	R/W
CurrentQuality	HmiQuality	Specifies the connection status of clock	True
DialBackColor	Color	Specifies the dial background color of clock	False
DialLabelColor	Color	Specifies the dial label color of clock	False
DialMode	HmiScaleMode	Specifies the dial mode of clock	False
DialTickColor	Color	Specifies the dial tick color of clock	False
Enabled	bool	Specifies if the allow operation control is enabled or not	False
Height	uint	Specifies the height of clock	False
Top	int	Specifies the Y- position of clock	False
Left	int	Specifies the X-position of clock	False
Name	string	Specifies the name of clock	False
VisualizeQuality	bool	Specifies the visual quality of clock	False
RotationCenterX	float	Specifies the X pivot point of clock	False
RotationCenterY	float	Specifies the Y pivot point of clock	False
ShowHours	bool	Specifies if the show hours is enabled or not	False
ShowMinutes	bool	Specifies if the show minutes is enabled or not	False
ShowSeconds	bool	Specifies if the show seconds is enabled or not	False
Opacity	float	Specifies the opacity of clock The value of Opacity should lie between 0 - 1	False
TimeSource	string	Specifies the time source of clock	R/W
Rotation	short	Specifies the rotation angle of clock	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of clock	R/W
Visiblility	bool	Specifies the visibility of clock	R/W
Width	uint	Specifies the width of clock	R/W
TabIndex	ushort	Specifies the tab index of clock	R/W
Title	HmiTextPart	Specifies the title of clock	R
ToolTipText	MultiLingualText	Specifies the tool tip text of clock	R/W
EventHandlers	HmiClockEventHandlerComposition	Specifies the event handle of clock	R

Property Name	Property Type	Description	Accessibility
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
DialLabelFont	HmiFontPart	Specifies the dial label font of clock	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of clock modify the following program code :

```
private void ClockPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiClock clock = screenitems.Create<HmiClock>("Default Value334554");
// To access basic properties of clock
//Name
var name = clock.Name;
clock.Name = "Default Value";
//AlternateBackColor
var altbackcolor = clock.AlternateBackColor;
clock.AlternateBackColor = Color.Beige;
//Height
var height = clock.Height;
clock.Height = 100;
// To access multilingual property of clock
//ToolTipText
var tooltip = clock.ToolTipText;
var tooltiptext = clock.ToolTipText.Items[0].Text;
clock.ToolTipText.Items[0].Text = <body><p>"TestforMultilingualProperty"</p></body>;
// To access other typical properties
//DialLabelFont
var dialfont = clock.DialLabelFont;
dialfont.Italic = false;
var fontname = dialfont.FontName;
dialfont.FontName = HmiFontName.SimSun;
var size = dialfont.Size;
dialfont.Size = 10.2f;
var stike = dialfont.StrikeOut;
dialfont.StrikeOut = true;
var underline = dialfont.Underline;
dialfont.Underline = false;
dialfont.Bold = true;
//ShowHours
var showhour = clock.ShowHours;
clock.ShowHours = false;
}
```

Accessing gauge properties (RT Unified)

Property

The following properties are supported in gauge screen items:

Property Name	Property Type	Description	Accessibility
AlternateBackColor	Color	Specifies the alternate background color of gauge	R/W
AlternateBorderColor	Color	Specifies the alternate border color of gauge	R/W
Authorization	string	Specifies the authorization of gauge	R
BackgroundColor	Color	Specifies the background color of gauge	R/W
BorderColor	Color	Specifies the border color of gauge	R/W
BorderWidth	byte	Specifies the border width of gauge	R/W
RelativeToOrigin	bool	Specifies if relative origin value of gauge is shown ot not	R/W
CurrentQuality	HmiQuality	Specifies the connection status of gauge	R
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
Title	HmiTextPart	Specifies the title of gauge	R
Font	HmiFontPart	Specifies the font type of gauge	R/W
Label	HmiTextPart	Specifies the label of gauge	R
Height	uint	Specifies the height of gauge	R/W
Top	int	Specifies the X position of gauge	R/W
Left	int	Specifies the Y position of gauge	R/W
Name	string	Specifies the name of gauge The length of the name character should lie betweenn 1 - 128 characters	R/W
NormalRangeColor	Color	Specifies the normal range color of gauge	R/W
OriginValue	double	Specifies the origin value of gauge	R/W
OutputFormat	string	Specifies the output format of gauge The value of ouput format should lie between 1 - 128	R/W
PeakIndicators	HmiPeakIndicator	Specifies the peak indicator of gauge	R/W
ProcessValue	string	Specifies the process value of gauge	R/W
ProcessValueIndicatorBackColor	Color	Specifies the background color of process value indicator	R/W
ProcessValueIndicatorForeColor	Color	Specifies the foreground color of process value indicator	R/W
ProcessValueIndicatorMode	HmiProcessIndicatorMode	Specifies the mode value of process value indicator	R/W
Rotation	short	Specifies the rotation angle of gauge	R/W
VisualizeQuality	bool	Specifies the connection quality of gauge	R/W
RotationCenterX	float	Specifies the X pivot point of gauge	R/W
RotationCenterY	float	Specifies the Y pivot point of gauge	R/W
Opacity	float	Specifies the opacity of gauge The value of Opacity should lie between 0 - 1	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of gauge	R/W

Property Name	Property Type	Description	Accessibility
ScaleBackground-Color	Color	Specifies the scale background color of gauge	R/W
ScaleForeground-Color	Color	Specifies the scale foreground color of gauge	R/W
TrendIndicatorColor	Color	Specifies the show trend indicator of gauge	R/W
Visible	bool	Specifies the visiblity of gauge	R/W
Width	uint	Specifies the width of gauge	R/W
TabIndex	ushort	Specifies the tab index of gauge	R/W
ShowTrendIndicator	bool	Specifies if the trend indicator of a gauge will be shown	R/W
ToolTipText	string	Specifies the tool tip text of gauge	R/W
EventHandlers	HmiGaugeEventHand-lerComposition	Specifies the event handler of gauge	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
CurvedScale	HmiCurvedScalePart	Specifies the scale of gauge	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of gauge modify the following program code :

```
private void GaugeScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiGauge gauge = screenitems.Create<HmiGauge>("Default Value334554");
//Name
var name = gauge.Name;
gauge.Name = "Default Value";
//RotationalCenterPlacement
var rotation = gauge.RotationCenterPlacement;
gauge.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;
//Width
var width = gauge.Width;
gauge.Width = 100;
//ToolTipText
var tooltip = gauge.ToolTipText;
var tooltiptext = gauge.ToolTipText.Items[0].Text;
gauge.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//Font
var title = gauge.Title;
title.Text.items[0].Text = "<body><p>teststing</p></body>";
title.Visible = false;
title.ForeColor = Color.Black;
var font = title.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.Name;
font.Name = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Weight = HmiFontWeight.Bold;
}
```

Accessing IO Field properties (RT Unified)

Property

The following properties are supported in IO Field properties screen items:

Property Name	Property Type	Description	Access
AlternateBackColor	Color	Specifies the alternate background color of IO field	R/W
Enabled	Bool	Specifies if the allow operation control is enabled or not	R/W

Property Name	Property Type	Description	Access
AlternateBorderColor	Color	Specifies the alternate border color of IO field	R/W
Authorization	String	Specifies the authorization of IO field	R
BackColor	Color	Specifies the background color of IO field	R/W
BorderColor	Color	Specifies the border color of IO field	R/W
BorderWidth	byte	Specifies the border width of IO field	R/W
CurrentQuality	HmiQuality	Specifies the connection status of IO field	R
Font	HmiFontPart	Specifies the font of IO field	R/W
ForegroundColor	Color	Specifies the foreground color of IO field	R/W
Height	uint	Specifies the height of IO field	R/W
HorizontalTextAlignment	HmiHorizontalAlignment	Specifies the horizontal alignment of IO Field	R/W
IOFieldType	HmiIOFieldType	Specifies the mode of IO field	R/W
Top	int	Specifies the Y position of IO field	R/W
Left	int	Specifies the X position of IO field	R/W
Rotation Angle	short	Specifies the rotation of IO field	R/W
VisualizeQuality	bool	Specifies the visualizing quality of IO field	R/W
TextTrimming	HmiTextTrimming	Specifies the text to be trimmed of IO field	R/W
RotationCenterX	float	Specifies the X pivot point of IO field	R/W
RotationCenterY	float	Specifies the Y pivot point of IO field	R/W
Opacity	float	Specifies the opacity of IO field The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of IO field The length of the name character should lie betweenn 1 - 128 characters	R/W
OutputFormat	string	Specifies the output format of IO field	R/W
Padding	HmiPaddingPart	Specifies the padding of IO field	R
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of IO field	R/W
ProcessValue	string	Specifies the process value of IO field	R/W
TabIndex	ushort	Specifies the tab index of IO field	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of IO field	R/W
VerticalTextAlignment	HmiVerticalAlignment	Specifies the vertical alignment of IO field	R/W
Visible	bool	Specifies the visiblity of IO field	R/W
Width	uint	Specifies the width of IO field	R/W
EventHandlers	HmiIOFieldEventHandle rComposition	Specifies the event handler of IO field	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
InputBehavior	HmiInputBehaviorPart	Specifies the reaction to input of IO field	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of IO Field screen items modify the following program code:

```
private void IOFieldScreenItemsPropertiesAccess(Project project)
{
  HmiSoftware hmiSoftware = GetHmiSoftware();
  HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
  HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
  HmiIOField iofield = screenitems.Create<HmiIOField>("Default Value354");
  //Name
  var name = iofield.Name;
  iofield.Name = "DefaultName";
  //AlternateBackColor
  var altbackcolor = iofield.AlternateBackColor;
  iofield.AlternateBackColor = Color.Beige;
  //Height
  var height = iofield.Height;
  iofield.Height = 100;
  //ToolTipText
  string culture = "en-US";
  Language lang = project.LanguageSettings.Languages.Find(new CultureInfo(culture));
  MultilingualText mltprop = ioField.ToolTipText;
  MultilingualTextItemComposition textItemComp = mltprop.Items;
  MultilingualTextItem mlttextitem = textItemComp.Find(lang);
  mlttextitem.Text = "CommentInEnglish";
  //Padding
  var padding = iofield.Padding;
  var bottom = padding.Bottom;
  padding.Bottom = 50;
  padding.Left = 60;
  padding.Right = 70;
  padding.Top = 50;
}
```

Accessing listbox properties (RT Unified)

Property

The following properties are supported in listbox screen items:

Property Name	Property Type	Description	Accessibility
AlternateBackgroundColor	Color	Specifies the alternative background color of listbox	R/W
AlternateBorderColor	Color	Specifies the alternative border color of listbox	R/W

Property Name	Property Type	Description	Accessibility
Authorization	string	Specifies the authorization of listbox	R
BackgroundColor	Color	Specifies the background color of listbox	R/W
BorderColor	Color	Specifies the border color of listbox	R/W
BorderWidth	byte	Specifies the border width of listbox	R/W
Content	HmiContentPart	Specifies the content of listbox	R/W
CurrentQuality	HmiQuality	Specifies the connection status of listbox	R
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
Font	HmiFontPart	Specifies the font part of listbox	R/W
ForeGroundColor	Color	Specifies the foreground color of listbox	R/W
Top	int	Specifies the X position of listbox	R/W
Left	int	Specifies the Y position of listbox	R/W
Name	string	Specifies the name of listbox The length of the name character should lie betweenn 1 - 128 characters	R/W
Process Value	string	Specifies the process value of listbox	R/W
Rotation	short	Specifies the rotation angle of listbox	R/W
VisualizeQuality	bool	Specifies the connection status of listbox	R/W
RotationCenterX	float	Specifies the X pivot point of listbox	R/W
RotationCenterY	float	Specifies the Y pivot point of listbox	R/W
SelectionItemHeight	uint16	Specifies the selected item height of listbox	R/W
SelectionItems	IList<IHmiSelectionItemPart>	Specifies the selected items of listbox	R
SelectionMode	HmiSelectionMode	Specifies the selected mode of listbox	R/W
SelectorPosition	HmiHorizontalAlignment	Specifies the selected position of listbox	R/W
Opacity	float	Specifies the opacity of listbox The value of Opacity should lie between 0 - 1	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of listbox	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Width	uint	Specifies the width of listbox	R/W

Property Name	Property Type	Description	Accessibility
TabIndex	ushort	Specifies the tab index of listbox	R/W
ToolTipText	string	Specifies the tool tip text of listbox	R/W
EventHandlers	HmiListBoxEventHandler Composition	Specifies the event handle of listbox	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
Padding	HmiPaddingPart	Specifies the padding of listbox	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of list box modify the following program code:

```
private void ListBoxPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiListBox listbox = screenitems.Create<HmiListBox>("Default Value334554");
//Name
var name = listbox.Name;
listbox.Name = "Default Value";
//AlternateBackColor
var altbackcolor = listbox.AlternateBackColor;
listbox.AlternateBackColor = Color.Beige;
//Height
var height = listbox.Height;
listbox.Height = 100;
// To access multilingual property
var tooltip = listbox.ToolTipText;
var tooltiptext = listbox.ToolTipText.Items[0].Text;
// To access other typical property of listbox
//Font
var font = listbox.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.Name;
font.Name = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Weight = HmiFontWeight.Bold;
//Selection Items
var selectionItem = listbox.SelectionItems;
var newselectionitem = selectionItem.Create();
var graphic = newselectionitem.Graphic;
newselectionitem.Graphic = "abcd";
newselectionitem.IsSelected = false;
}
```

Accessing radio button properties (RT Unified)

Property

The following properties are supported in radio button screen items:

Property Name	Property Type	Description	Accessibility
AlternateBackground-Color	Color	Specifies the alternative background color of radio button	R/W
AlternateBorderColor	Color	Specifies the alternative border color of radio button	R/W
Authorization	string	Specifies the authorization of radio button	R
BackgroundColor	Color	Specifies the background color of radio button	R/W
BorderColor	Color	Specifies the border color of radio button	R/W
BorderWidth	byte	Specifies the border width of radio button	R/W
Content	HmiContentPart	Specifies the content of radio button	R/W
CurrentQuality	HmiQuality	Specifies the connection status of radio button	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
Font	HmiFontPart	Specifies the font of radio button	False
ForegroundColor	Color	Specifies the foreground color of radio button	R/W
Height	uint	Specifies the height of radio button	R/W
Top	int	Specifies the X position of radio button	R/W
Left	int	Specifies the Y position of radio button	R/W
Name	string	Specifies the name of radio button The length of the name character should lie betweenn 1 - 128 characters	R/W
Process Value	string	Specifies the process value of radio button	R/W
Rotation	short	Specifies the rotation angle of radio button	R/W
VisualizeQuality	bool	Specifies the connection status of radio button	R/W
RotationCenterX	float	Specifies the X pivot point of radio button	R/W
RotationCenterY	float	Specifies the Y pivot point of radio button	R/W
SelectionItemHeight	uint16	Specifies the selection item height of radio button	R/W
SelectionItems	IList<IHmiSelectionItemPart>	Specifies the selection items of radio button	R
SelectorPosition	HmiHorizontalAlignment	Specifies the selector position of radio button	R
Opacity	float	Specifies the opacity of radio button The value of Opacity should lie between 0 - 1	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of radio button	R/W

Property Name	Property Type	Description	Accessibility
Visible	bool	Specifies if the visible is enabled or not	R/W
Width	uint	Specifies the width of radio button	R/W
TabIndex	ushort	Specifies the tab index of radio button	R/W
ToolTipText	string	Specifies the tool tip text of radio button	R/W
EventHandlers	HmiRadioradio buttonGroup EventHandlerComposition	Specifies the event handle of radio button	R
Padding	HmiPaddingPart	Specifies the padding of radio button	R
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of radio button modify the following program code:

```
private void RadiobuttonPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiRadioButtonGroup radio = screenitems.Create<HmiRadioButtonGroup>("Default Value334554");
// To access basic properties
//Name
var name = radio.Name;
radio.Name = "Default Value";
//AlternateBackColor
var altbackcolor = radio.AlternateBackColor;
radio.AlternateBackColor = Color.Beige;
//Height
var height = radio.Height;
radio.Height = 100;
// To access multilingual property
//ToolTipText
var tooltip = radio.ToolTipText;
var tooltiptext = radio.ToolTipText.Items[0].Text;
//radio.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
// To access other typical properties of radio button
//Font
var font = radio.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.Name;
font.Name = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Weight = HmiFontWeight.Bold;
//Selection Items
var selectionItem = radio.SelectionItems;
var newselectionitem = selectionItem.Create();
var graphic = newselectionitem.Graphic;
newselectionitem.Graphic = "abcd";
newselectionitem.IsSelected = false;
}
```

Accessing slider properties (RT Unified)

Property

The following properties are supported in slider screen items:

Property Name	Property Type	Description	Accessibility
AlternativeBackColor	Color	Specifies the alternative background color of slider	R/W
AlternativeBorderColor	Color	Specifies the alternative border color of slider	R/W
Authorization	string	Specifies the authorization of slider	R
BackgroundColor	Color	Specifies the background color of slider	R/W
BorderColor	Color	Specifies the border color of slider	R/W
BorderWidth	byte	Specifies the border width of slider	R/W
BarMode	HmiBarMode	Specifies the bar mode of slider	R/W
RelativeToOrigin	bool	Specifies the relative to origin value of slider	R/W
CurrentQuality	HmiQuality	Specifies the connection status of slider	R
Enabled	bool	Specifies if the allow operation control is enabled or not	False
Title	HmiTextPart	Specifies the title of slider	R
Font	HmiFontPart	Specifies the font type of slider	R/W
Label	HmiTextPart	Specifies the label of slider	R
Height	uint	Specifies the height of slider	R/W
Top	int	Specifies the X position of slider	R/W
Left	int	Specifies the Y position of slider	R/W
Name	string	Specifies the name of slider The length of the name character should lie betweenn 1 - 128 characters	R/W
NormalRangeColor	Color	Specifies the normal range color of slider	R/W
OriginValue	double	Specifies the origin value of slider	R/W
OutputFormat	string	Specifies the output format of slider The value of ouput format should lie between 1 - 128	R/W
PeakIndicators	HmiPeakIndicator	Specifies the peak indicator of slider	R/W
Process Value	string	Specifies the process value of slider	R/W
ProcessValueIndicatorBackColor	Color	Specifies the background color process value indicator of slider	R/W
ProcessValueIndicatorForeColor	Color	Specifies the foreground color process value indicator of slider	R/W
ProcessValueIndicatorMode	HmiProcessIndicatorMode	Specifies the mode of process value indicator of slider	R/W

Property Name	Property Type	Description	Accessibility
Rotation	short	Specifies the rotation angle of slider	R/W
VisualizeQuality	bool	Specifies the connection quality of slider	R/W
RotationCenterX	float	Specifies the X pivot point of slider	R/W
RotationCenterY	float	Specifies the Y pivot point of slider	R/W
Opacity	float	Specifies the opacity of slider The value of Opacity should lie between 0 - 1	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of slider	R/W
ScaleBackgroundColor	Color	Specifies the scale background color of slider	R/W
ScaleForegroundColor	Color	Specifies the scale foreground color of slider	R/W
TrendIndicatorColor	bool	Specifies the show trend indicator of slider	R/W
Visible	bool	Specifies the visiblity of slider	R/W
Width	uint	Specifies the width of slider	R/W
ValuePosition	HmiSimplePosition	Specifies the value position of slider	R/W
WriteDuringChange(Write process value Immediatly)	bool	Specifies if the write process value is enabled or not	R/W
TabIndex	ushort	Specifies the tab index of slider	R/W
ShowTrendIndicator	bool	Specifies the trend indicator color of slider	R/W
ToolTipText	string	Specifies the tool tip text of slider	R/W
ShowValue	bool	Specifies if the value of a slider will be shown	R/W
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
EventHandlers	HmiSliderEventHandlerComposition	Specifies the event handler of slider	R
ThumbBackColor	color	Specifies the thumb background color of slider	R/W
ThumbForeColor	color	Specifies the thumb foreground color of slider	R/W
StraightScale	HmiStraightScalePart	Specifies the straight scale of slider	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of slider modify the following program code :

```
private void SliderPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiSlider slider = screenitems.Create<HmiSlider>("Default Value334554");
// To access basic properties of slider
//Name
var name = slider.Name;
slider.Name = "Default Value";
//RotationalCenterPlacement
var rotation = slider.RotationCenterPlacement;
slider.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;
//Width
var width = slider.Width;
slider.Width = 100;
// To access multilingual property of slider
//ToolTipText
var tooltip = slider.ToolTipText;
var tooltiptext = slider.ToolTipText.Items[0].Text;
//slider.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
// To access other typical properties of slider
//Font
var title = slider.Title;
//title.Text.Items[0].Text = "teststing";
title.Visible = false;
title.ForeColor = Color.Black;
var font = title.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.Name;
font.Name = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Weight = HmiFontWeight.Bold;
}
```

Accessing switch properties (RT Unified)

Property

The following properties are supported in switch screen items:

Property Name	Property Type	Description	Access
AlternateBackgroundColor	Color	Specifies the alternate background color of switch	R/W
AlternateBorderColor	Color	Specifies the alternate border color of switch	R/W
IsAlternateState	bool	Specifies the alternate state of switch	R/W
Authorization	string	Specifies the authorization of switch	R
BackgroundColor	Color	Specifies the background color of switch	R/W
BorderColor	Color	Specifies the border color of switch	R/W
BorderWidth	byte	Specifies the border width of switch	R/W
Contents	HmiContentPart	Specifies the content of switch	R/W
CurrentQuality	HmiQuality	Specifies the connection status of switch	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
Font	HmiFontPart	Specifies the font of switch	R/W
ForegroundColor	Color	Specifies the foreground color of switch	R/W
Height	uint	Specifies the height of switch	R/W
Top	int	Specifies the Y position of switch	R/W
Left	int	Specifies the X position of switch	R/W
Rotation	short	Specifies the rotation of switch	R/W
Text	MultiLingualText	Specifies the text of switch	R
VisualizeQuality	bool	Specifies the visualize quality of switch	R/W
RotationCenterX	float	Specifies the X pivot point of switch	R/W
RotationCenterY	float	Specifies the Y pivot point of switch	R/W
Opacity	float	Specifies the opacity of switch The value of Opacity should lie between 0 - 1	R/W
Name	string	Specifies the name of switch The length of the name character should lie betweenn 1 - 128 characters	R/W
Padding	HmiPaddingPart	Specifies the padding of switch	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of switch	R/W
TabIndex	ushort	Specifies the tab index of switch	R/W
ToolTipText	MultilingualText	Specifies the tool tip text of switch	R/W
Visiblity	bool	Specifies the visiblity of switch	R/W
Width	uint	Specifies the width of switch	R/W

Property Name	Property Type	Description	Access
EventHandlers	HmiToggleswitchEventHandlerComposition	Specifies the event handler of switch	R/W
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
Graphic	string	Specifies the graphic of switch The value of graphic should lie between 1 - 128	R/W
AlternateGraphic	string	Specifies the graphic with pressed switch The value of Alternate Graphic should lie between 1 - 128	R/W
AlternateText	MultiLingualText	Specifiies the text with pressed switch	R
Padding	HmiPaddingPart	Specifiies the padding of switch	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of switch screen items modify the following program code:

```
private void switchScreenItemsPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.ScreenItems;
HmiToggleSwitch toggleswitch = screenitems.Create<HmiToggleSwitch>("Default Value354");
//Name
var name = toggleswitch.Name;
toggleswitch.Name = "Default Value";
//AlternateBackColor
var altbackcolor = toggleswitch.AlternateBackColor;
toggleswitch.AlternateBackColor = Color.Beige;
//Height
var height = toggleswitch.Height;
toggleswitch.Height = 100;
//ToolTipText
var tooltip = toggleswitch.ToolTipText;
var tooltiptext = toggleswitch.ToolTipText.Items[0].Text;
toggleswitch.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
//Font
var font = toggleswitch.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.Name;
font.Name = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Weight = HmiFontWeight.Bold;
}
```

Accessing symbolic IOField properties (RT Unified)

Property

The following properties are supported in HMI Symbolic IO Field:

Property Name	Property Type	Description	Access
Authorization	string	Specifies the authorization of symbolic IO field	R/W
CurrentQuality	HmiQuality	Specifies the connection quality of symbolic IO field	R
Dynamizations	DynamizationBaseComposition	Specifies the dynamizations of symbolic IO field	R

Property Name	Property Type	Description	Access
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
EventHandlers	HmiSymbolicIOFieldEventHandlerComposition	Specifies the event handle of symbolic IO field	R
Height	uint	Specifies the height of symbolic IO field	R
IOFieldType	HmiIOFieldType	Specifies the field type of symbolic IO field	R/W
ProcessValue	String	Specifies the process value of symbolic IO field	R/W
ResourceList	String	Specifies the textlist and graphic list for symbolic IO field	R/W
Selected Index	Int32	Specifies the selected index of symbolic IO field	R
SelectionBackColor	Color	Specifies the selected background color of symbolic IO field	R/W
SelectionForeColor	Color	Specifies the selected foreground color of symbolic IO field	R/W
Text	MultilingualText	Specifies the text of symbolic IO field	R/W
Left	int	Specifies the X-axis position of symbolic IO field	R/W
Name	String	Specifies the name of symbolic IO field	R/W
Opacity	float	Specifies the opacity value of symbolic IO field	R/W
PropertyEventHandlers	PropertyEventHandlerComposition	Specifies the property event handlers of symbolic IO field	R
RequireExplicitUnlock	bool	Specifies if the value of explicit unlock is shown or not	R/W
RotationAngle	short	Specifies the rotation angle of symbolic IO field	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of symbolic IO field	R/W
RotationCenterX	float	Specifies the X pivot point of symbolic IO field	R/W
RotationCenterY	float	Specifies the Y pivot point of symbolic IO field	R/W
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W
TabIndex	ushort	Specifies the tab index of checkbox	R/W
ToolTipText	MultilingualText	Specifies the text to be mentioned on tool tip	R
Top	int	Specifies the Y-position of symbolic IO field	R/W
Visible	bool	Specifies the visibility of symbolic IO field	R/W

Property Name	Property Type	Description	Access
VisualizeQuality	bool	Specifies the visual quality of symbolic IO field	R/W
Width	uint	Specifies the width of symbolic IO field	R/W

Requirement

- The HMI Software Object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the basic properties of Symbolic IO Field modify the following program code

```
private void SymbolicIOFieldPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreenComposition screens  =  hmiSoftware.Screens;
HmiScreen hmiScreen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenItems = hmiScreen.ScreenItems
HmiSymbolicIOField objSymbField =
screenItems.Create<HmiSymbolicIOField>("Screen_object_1");
//Left
objSymbField.Left = 1;
//ToolTipText
objSymbField.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></
body>";
}
```

Accessing touch area properties (RT Unified)

Property

The following properties are supported in HMI Touch Area:

Property Name	Property Type	Description	Accessibility
Authorization	string	Specifies the authorization of hmi touch area	R/W
Dynamizations	DynamizationBaseComposition	Specifies the dynamization of hmi touch area	R
Enabled	bool	Specifies if the allow operator control is enabled or not	R/W
EventHandlers	HmiTouchAreaEventHandlerComposition	Specifies the event handler of hmi touch area	R
Height	uint	Specifies the height of hmi touch area	R/W

Property Name	Property Type	Description	Accessibility
BackColor	Color	Specifies the background color of hmi touch area	R/W
Left	int	Specifies the X position of hmi touch area	R/W
Name	string	Specifies the name of hmi touch area	R/W
PropertyEventHandlers	PropertyEventHandlerComposition	Specifies the property event handler of hmi touch area	R
RequireExplicitUnlock	bool	Specifies if explicit unlock is required or not	R/W
TabIndex	ushort	Specifies the tab index of hmi touch area	R/W
Top	int	Specifies the Y position of hmi touch area	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Width	uint	Specifies the width of hmi touch area	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access basic properties of HMI Touch Area modify the following program code:

```
private void HMITouchAreaPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreenComposition screens = hmiSoftware.Screens;
HmiScreen screen = hmiSoftware.Screens.Find("Screen_1");
HmiScreenItemBaseComposition screenItems = hmiScreen.ScreenItems;
HmiTouchArea objTouchArea = screenItems.Create<HmiTouchArea>("Screen_object_1");
//Left
objTouchArea.Left = 1;
//Backcolor
objTouchArea.Backcolor = Color.Red;
}
```

Controls (RT Unified)

Accessing alarm control properties (RT Unified)

roperty

The following properties are supported in alarm control screen items.

Property Name	Property Type	Description	Accessibility
UseAlarmColors	bool	Specifies the user alarm color of alarm control	R/W
Filter	string	Specifies the filter of alarm control	R/W
AlwaysShowRecent	bool	Specifies the show recent of alram control	R/W
AlarmSourceType	HmiAlarmSource-Type	Specifies the alarm source of alarm control	R/W
Caption	HmiTextPart	Specifies the label to be displayed on alarm control	R
WindowFlags	HmiWindowFlag	Specifies the window setting of alarm control	R/W
Icon	string	Specifies the icon of alam control	R/W
Top	int	Specifies the Y Position of alarm control	R/W
Left	int	Specifies the X Position of alarm control	R/W
Width	uint	Specifies the width of alarm control	R/W
Height	uint	Specifies the height of alarm control	R/W
CurrentQuality	HmiQuality	Specifies the connection status of alarm control	R
Name	string	Specifies the name of alarm control	R/W
Visible	bool	Specifies the visibility of alarm control	R/W
Enabled	bool	Specifies if the allow operator control	R/W
TabIndex	ushort	Specifies the tab index of alarm control	R/W
ToolBar	HmiToolBarPart	Specifies the toolbar of alarm control	R
StatusBar	HmiStatusBarPart	Specifies the status bar of alarm control	R
EditMode	HmiEditMode	Specifies the editing mode of alarm control	R/W
TimeZone	int	Specifies the time zone of alarm control	R/W
AlarmDefinitionView-Setup	HmiVisibleAlarms	Specifies the displayed alarms of alarm control	R/W
ActiveAlarmsView-Setup	HmiVisibleAlarms	Specifies the current alarms of alarm control	R/W
SuppressFlashing	bool	Specifies the suppress flashing of alarm control	R/W
AcknowledgmentFlashingRate	HmiFlashingRate	Specifies the flashing rate acknowledgement messages of alarm control	R/W
ResetFlashingRate	HmiFlashingRate	Specifies the reset flashing rate of alarm control	R/W
DefaultSortDirection	HmiSortDirection	Specifies the default sorting direction	R/W
BackColor	Color	Specifies the background color of alarm control	R/W
AlarmView	HmiDataGridView-Part	Specifies the alarm view of alarm control	TRUE

Property Name	Property Type	Description	Accessibility
EventHandlers	HmiAlarmControlEventHandlerComposition	Specifies the event handlers of alarm control	R
CaptionColor	Color	Specifies the caption color of alarm control	R/W
AlarmStatisticsView	HmiDataGridViewPart	Specifies the alarm statistic view	R
ShowFocusVisual	Boolean	Specifies the show focus visual of alarm control	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of alarm control modify the following program code:

```
private void AccessAlarmControlProperties()
{
  HmiSoftware hmiSoftware = GetHmiSoftware();
  var screen = hmiSoftware.Screens;
  var createdscreen = screen.Create("TestScreen_1");
  var hmialarmcontrol=
Createdscreen.ScreenItems.Create<HmiAlarmControl>("HmiAlarmControl_1");
  // Basic properties Access
  // Width
  var autoplay = hmialarmcontrol.Width;
  hmialarmcontrol.Width = 100;
  // BackColor
  var backcolor = hmialarmcontrol.BackColor;
  hmialarmcontrol.BackColor = Color.Beige;
  // Name
  var name = hmialarmcontrol.Name;
  hmialarmcontrol.Name = "DefaultName";
  // MultiLingual Property Access
  // Access caption
  var caption = hmialarmcontrol.Caption;
  IHmiWindowFeature.Caption.Text.Items[0].Text = "<body><p>TestforMultilingualProperty</
body></p>";
  // Access other typical properties
  var statusBar = hmialarmcontrol.StatusBar;
  var backColor = statusBar. BackColor;
     statusBar.BackColor = Color.Aqua;
  var enabled = statusBar.Enabled;
  statusBar.Enabled = true;

     var visible = statusBar. Visible;
  statusBar.Visible = true;
}
```

Accessing function trend control properties (RT Unified)

Property

The following properties are supported in function function trend control.

Property Name	Property Type	Description	Accessibility
ShowStatisticRulers	bool	Specifies the show statistic rulers of function trend control	R/W
AreaSpacing	ushort	Specifies the area space on function trend control	R/W
Online	bool	Specifies the online	R
ExtendRulerToAxis	bool	Specifies the extend ruler to axis on function trend control	R/W
Caption	MultilingualProperty	Specifies the text to be shown in the caption of a screen window or windowed control (Label)	R/W
WindowFlags	HmiWindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W
Icon	string	Specifies the icon on the control window	R/W
Top	Int	Specifies the value of Y- coordinates of function trend control	R/W
Left	int	Specifies the value of X- Coordinate of function trend control	R/W
Width	uint	Specifies the width value of function trend control	R/W
Height	uint	Specifies the height value of function trend control	R/W
CurrentQuality	HmiQuality	Specifies the connection status of function trend control	R
Name	string	Specifies the name of function trend control	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Enabled	bool	Specifies if allow operator control is enabled or not	R/W
TabIndex	ushort	Screen items specifying a tab index of 0 are not part of the tab order	R/W
ToolBar	HmiToolBarPart	Specifies the tool bar of function trend control	R
StatusBar	HmiStatusBarPart	Specifies the status bar on function trend control	R
	HmiFontPart	Specifies the font type of function trend control	R
TimeZone	int	Specifies the time zone	R/W
ShowRuler	bool	Specifies the ruler	False
ShiftAxes	bool	Specifies the shift axes of function trend control	R/W
BackColor	Color	Specifies the background color of function trend control	R/W

Property Name	Property Type	Description	Accessibility
Legend	HmiLegendPart	Specifies the legend	R
EventHandlers	HmiFunctionTrend-ControlEventHandler-Composition	Specifies the event handlers of function trend control	R
FunctionTrendAreas	HmiFunctionTrendAr-eaPartComposition	Specifes the function trend areas	R
CaptionColor	Color	Specifies the caption color of function trend control	R/W
ShowFocusVisual	bool	Specifies if the show focus is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of function function trend control modify the following program code:

```
private void FunctionTrendControlPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen screen = hmiSoftware.Screens.Create("Screen_164");
HmiFunctionTrendControl hmifunctiontrendcontrol =
screen.ScreenItems.Create<HmiFunctionTrendControl>("CTrendControl");
// To access basic properties of function function trend control
//Width
var autoplay = hmifunctiontrendcontrol.Width;
hmifunctiontrendcontrol.Width = 100;
//BackColor
var backcolor = hmifunctiontrendcontrol.BackColor;
hmifunctiontrendcontrol.BackColor = Color.Beige;
//Name
var name = hmifunctiontrendcontrol.Name;
hmifunctiontrendcontrol.Name = "DefaultName";
// To acccess multilingual property
//Caption
var caption = hmifunctiontrendcontrol.Caption.Items[0].Text;
hmifunctiontrendcontrol.Caption.Items[0].Text =
"<body><p>TestforMultilingualProperty</p></body>";
// To access other typical properties
//Status bar
var statusBar = hmifunctiontrendcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
}
```

Program code: TrendArea in FunctionTrendControl using part

To create a trendarea in a Hmi Function function trend control using part modify the following program code:

```
private void TrendAreaCreateUsingPart()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
var screen = hmiSoftware.Screens;
var createdscreen = screen.Create("TestScreen_1");
HmiFunctionTrendControl hmifunctiontrendcontrol =
hmiSoftware.Screens[0].ScreenItems.Create<HmiFunctionTrendControl>("CTrendControl_1");
HmiFunctionTrendAreaPart part = hmifunctiontrendcontrol.FunctionTrendAreas.Create("part1");
}
```

To search a trendarea in Hmi Function function trend control using part modify the following program code :

```
private void TrendAreaPartSearch()
{
HmiFunctionTrendAreaPart part = hmifunctiontrendcontrol.FunctionTrendAreas.Find("part1");
}
```

To check a trendarea in Hmi Function function trend control using part modify the following program code :

```
private void IsTrendAreaExist()
{
bool bPresent = hmifunctiontrendcontrol.FunctionTrendAreas.Contains(part);
}
```

Accessing media player properties (RT Unified)

Introduction

The following properties are supported in media player screen items.

Property Name	Property Type	Description	Accessibility
Url	string	Specifies the url of media player	R/W
AutoPlay	bool	Specifies the auto play of media player	R/W
VideoOutput	HmiVideoOutput	Specifies the video output of media player	R/W
BackColor	Color	Specifies the background color of radio process control	R/W
Caption	MultilingualProperty	Specifies the caption of process control	R/W
WindowFlags	HmiWindowFlag	Specifies the window flags of process control	R/W
Icon	string	Specifies the icon of process control	R/W
Top	int	Specifies the X position of process control	R/W

Property Name	Property Type	Description	Accessibility
Left	int	Specifies the Y position of process control	R/W
Width	uint	Specifies the width of process control	R/W
Height	uint	Specifies the height of process control	R/W
CurrentQuality	HmiQuality	Specifies the connection status of media player	True
Name	string	Specifies the name of process control The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
TabIndex	ushort	Specifies the tab index of process control	R/W
ToolBar	HmiToolBarPart	Specifies the tool bar of process control	R
StatusBar	HmiStatusBarPart	Specifies the status bar of process control	R
EventHandlers	HmiMediaControlEventHandlerComposition	Specifies the event handle of process control	R
CaptionColor	Color	Specifies the caption color of process control	R/W
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access basic properties of media player modify the following program code:

```
private void MediaPlayerPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
var screen = hmiSoftware.Screens;
var createdscreen = screen.Create("TestScreen_156");
var hmimediacontrol =
createdscreen.ScreenItems.Create<HmiMediaControl>("HmiMediaControl_1");
// To access basic properties
//Autoplay
var autoplay = hmimediacontrol.AutoPlay;
hmimediacontrol.AutoPlay = true;
//BackColor
var backcolor = hmimediacontrol.BackColor;
hmimediacontrol.BackColor = Color.Beige;
//Name
var name = hmimediacontrol.Name;
hmimediacontrol.Name = "DefaultName";
// To access multilingual properties
//Caption
var caption = hmimediacontrol.Caption.Items[0].Text;
hmimediacontrol.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
// To access other typical propeties
var statusBar = hmimediacontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
}
```

Accessing process control properties (RT Unified)

Property

The following properties are supported in process control screen items:

Property Name	Property Type	Description	Accessibility
Online	bool	Specifies the online connection of process control	R/W
TimeStepSmoothing-Base	HmiTimeRangeBase	Specifies the time interval for smoothing base	R/W
TimeStepSmoothingFac-tor	int	Specifies the time interval for smoothing factor	R/W
BackColor	Color	Specifies the background color of process control	R/W
Caption	MultilingualProperty	Specifies the text mentioned on process control	R/W

Property Name	Property Type	Description	Accessibility
WindowFlags	HmiWindowFlag	Specifies the window flags of process control	R/W
Icon	string	Specifies the icon of process control	R/W
Top	int	Specifies the X position of process control	R/W
Left	int	Specifies the Y position of process control	R/W
Width	uint	Specifies the width of process control	R/W
Height	uint	Specifies the height of process control	R/W
CurrentQuality	HmiQuality	Specifies the connection status of process control	R
Name	string	Specifies the name of process control The length of the name character should lie betweenn 1 - 128 characters	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Enabled	bool	Specifies if the allow operation control is enabled or not	R/W
TabIndex	ushort	Specifies the tab index of process control	R/W
ToolBar	HmiToolBarPart	Specifies the tool bar of process control	R
StatusBar	HmiStatusBarPart	Specifies the status bar of process control	R
EditMode	HmiEditMode	Specifies the edit mode of process control	R/W
TimeZone	int	Specifies the time zone of process control	R/W
ProcessView	HmiDataGridViewPart	Specifies the process view of process control	R
EventHandlers	HmiProcessControlEventHandlerComposition	Specifies the event handle of process control	R
CaptionColor	Color	Specifies the caption color of process control	R/W
ShowFocusVisual	bool	Specifies if the show focus visual is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of process control modify the following program code :

```
private void ProcessControlPropertiesAccess()
{
// To access basic properties
//Width
var autoplay = hmiprocesscontrol.Width;
hmiprocesscontrol.Width = 1;
//BackColor
var backcolor = hmiprocesscontrol.BackColor;
hmiprocesscontrol.BackColor = Color.Beige;
//Name
var name = hmiprocesscontrol.Name;
hmiprocesscontrol.Name = "DefaultName";
// To access multilingual property
//Caption
var caption = hmiprocesscontrol.Caption.Items[0].Text;
hmiprocesscontrol.Caption.Items[0].Text= "<body><p>TestforMultilingualProperty</p></
body>";
// To access other typical properties
//status bar
var statusBar = hmiprocesscontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
}
```

Accessing parameter set control properties (RT Unified)

Property

The following properties are supported in parameter set control:

Property Name	Property Type	Description	Accessibility
ParameterSetType-Fixed	string		False
BackColor	Color	Specifies the background color of parameter set control	False
Caption	MultilingualProperty	Specifies the text to be shown in the caption of a screen window or windowed control (Label)	False
WindowFlags	HmiWindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	False
Icon	string	Specifies the icon on the control window	False

Property Name	Property Type	Description	Accessibility
Top	int	Specifies the value of Y- coordinates of parameter set control	False
Left	int	Specifies the value of X- Coordinate of parameter set control	False
Width	uint	Specifies the width value of parameter set control	False
Height	uint	Specifies the height value of parameter set control	False
CurrentQuality	HmiQuality	Specifies the connection status of parameter set control	True
Name	string	Specifies the name of parameter set control	False
Visible	bool	Specifies if the visible is enabled or not	False
Enabled	bool	Specifies if allow operator control is enabled or not	False
TabIndex	ushort	Screen items specifying a tab index of 0 are not part of the tab order	False
ToolBar	HmiToolBarPart	Specifies the tool bar of parameter set control	True
StatusBar	HmiStatusBarPart	Specifies the status bar on parameter set control	True
EditMode	HmiEditMode	Specifies the edit mode of parameter set control	False
TimeZone	int	Specifies the time zone of parameter set control	False
ParameterView	HmiDataGridViewPart	Specifies the parameter view of parameter set control	True
ForeColor	Color	Specifies the foreground color of parameter set control	False
SelectionBackColor	Color	Specifies the selected background color of parameter set control	False
SelectionForeColor	Color	Specifies the selected foreground color of parameter set control	False
Font	HmiFontPart	Specifies the font of parameter set control	True
EventHandlers	HmiDetailedParameterControlEventHandlerComposition	Specifies the event handler of parameter set control	True
CaptionColor	Color	Specifies the caption color of parameter set control	False
HideDetails	bool	Specifies if hide details is enabled or not	False
ShowFocusVisual	bool	Specifies if the show focus is enabled or not	False

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of parameter set control modify the following program code :

```
private void ParameterSetControlPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen screen = hmiSoftware.Screens.Create("TestScreen");
HmiDetailedParameterControl hmidetailedparametercontrol =
screen.ScreenItems.Create<HmiDetailedParameterControl>("ParameterControl_1");
// To access basic properties
//Width
var autoplay = hmidetailedparametercontrol.Width;
hmidetailedparametercontrol.Width = 100;
//BackColor
var backcolor = hmidetailedparametercontrol.BackColor;
hmidetailedparametercontrol.BackColor = Color.Beige;
//Name
var name = hmidetailedparametercontrol.Name;
hmidetailedparametercontrol.Name = "Default";
// To access multilingual property
//Caption
var caption = hmidetailedparametercontrol.Caption.Items[0].Text;
hmidetailedparametercontrol.Caption.Items[0].Text=
"<body><p>TestforMultilingualProperty</p></body>";
// To access other typical property
//Status bar
var statusBar = hmidetailedparametercontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
}
```

Accessing screen window properties (RT Unified)

Property

The following properties are supported in screen window screen items:

Property Name	Property Type	Description	Accessibility
TabIntoWindow	bool	Specifies the tab into window of screen window	R/W
Screen	string	Specifies the screen of screen window	R/W
ScreenName	string	Specifies the screen name of screen window	R
ScreenNumber	byte	Specifies the screen number of screen window	R
System	byte	Specifies the system of screen window	R/W

Property Name	Property Type	Description	Accessibility
CurrentZoomFactor	double	Specifies the current zoom factor of screen window	R/W
VerticalScrollBarVisibility	HmiScrollBarVisibility	Specifies the vertical scroll bar visibility of screen window	R/W
VerticalScrollBarPosition	int	Specifies the vertical scroll bar position of screen window	R/W
HorizontalScrollBarVisibility	HmiScrollBarVisibility	Specifies the horizontal scroll bar visibility of screen window	R/W
HorizontalScrollBarPosition	int	Specifies the horizontal scroll bar position of screen window	R/W
Adaption	HmiScreenWindowAdaption	Specifies the adaption of screen window	R/W
InteractiveZooming	bool	Specifies the interactive zooming of screen window	R
Caption	MultilingualProperty	Specifies the caption of screen window	R/W
WindowFlags	HmiWindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W
Icon	string	Specifies the icon of screen window	R/W
Top	int	Specifies the value of the Y coordinates of screen window	R/W
Left	int	Specifies the value of the X coordinates of screen window	R/W
Width	uintz	Specifies the width of screen window	R/W
Height	uint	Specifies the height of screen window	R/W
CurrentQuality	HmiQuality	Specifies the connection status of screen window	R
Name	string	Specifies the name of screen window	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Enabled	bool	Specifies if allow operator control is enabled or not	R/W
TabIndex	ushort	Screen items specifying a tab index of 0 are not part of the tab order	R/W
EventHandlers	HmiScreenWindowEventHandlerComposition	Specifies the event handlers of screen window	R
CaptionColor	Color	Specifies the caption color of screen window	R/W
ShowFocusVisual	bool	Specifies if the show focus is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access properties of screen window modify the following program code:

```
private void ScreenWindowPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
var screen = hmiSoftware.Screens;
var createdscreen = screen.Create("TestScreen_15666");
var hmiscreenwindow =
createdscreen.ScreenItems.Create<HmiScreenWindow>("HmiScreenWindow_1")
// To access basis property of screen window
//Width
var autoplay = hmiscreenwindow.Width;
hmiscreenwindow.Width = 100;
//HorizontalScrollBarVisibility
var horizontalscrollbarvisibilityValue = hmiscreenwindow.HorizontalScrollBarVisibility;
hmiscreenwindow.HorizontalScrollBarVisibility = HmiScrollBarVisibility.Visible;
//Name
var name = hmiscreenwindow.Name;
hmiscreenwindow.Name = "DefaultName";
// To access multilingual property
// Caption
hmiscreenwindow.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
}
```

Accessing system diagnosis control properties (RT Unified)

Property

The following properties are supported in system diagnosis screen items:

Property Name	Property Type	Description	Accessibility
SystemDiagnosisView	HmiDataGridView-Part	Specifies the system diagnosis view	R
Caption	HmiTextPart	Specifies the text to be shown in the caption of a screen window or windowed control (Label)	R
BackColor	Color	Specifies the background color of system diagnosis	R/W
WindowFlags	HmiWindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W
Icon	string	Specifies the icon on the system diagnosis	R/W
Top	int32	Specifies the value of Y- coordinates of system diagnosis	R/W
Left	int32	Specifies the value of X- Coordinate of system diagnosis	R/W
Width	uint32	Specifies the width value of system diagnosis	R/W

Property Name	Property Type	Description	Accessibility
Height	uint32	Specifies the height value of system diagnosis	R/W
CurrentQuality	HmiQuality	Specifies the connection status of system diagnosis	R
Name	string	Specifies the name of system diagnosis	R/W
ToolBar	HmiToolBarPart	Specifies the tool bar of system diagnosis	R
StatusBar	HmiStatusBarPart	Specifies the status bar on system diagnosis	R
TimeZone	int	Specifies the time zone of system diagnosis	R/W
CaptionColor	Color	Specifies the caption color of system diagnosis	R/W
ShowFocusVisual	bool	Specifies if show focus visual is enabled or not	R/W

Requirement

- The HMI Software objec is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of system diagnosis modify the following program code:

```
private void SystemDiagnosisControlPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen screen = hmiSoftware.Screens.Create("Screen_1");
HmiSystemDiagnosisControl syscontrol =
screen.ScreenItems.Create<HmiSystemDiagnosisControl>("SysDiag_1");
syscontrol.SystemDiagnosisView.RowHeight = 50;
Debug.Assert(syscontrol.SystemDiagnosisView.RowHeight.Equals(50));
syscontrol.TimeZone = 28;
Debug.Assert(syscontrol.TimeZone.Equals(28));
}
```

Accessing trend control properties (RT Unified)

Property

The following properties are supported in trend control screen items.

Property Name	Property Type	Description	Accessibility
ShowStatisticRulers	bool	Specifies the show statistic rulers of trend control	R/W
AreaSpacing	ushort	Specifies the area space on trend control	R/W
ExtendRulerToAxis	bool	Specifies the extend ruler to axis on trend control	R/W

Property Name	Property Type	Description	Accessibility
Caption	MultilingualProperty	Specifies the text to be shown in the caption of a screen window or windowed control (Label)	R/W
WindowFlags	HmiWindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W
Icon	string	Specifies the icon on the control window	R/W
Top	int	Specifies the value of Y- coordinates of trend control	R/W
Left	int	Specifies the value of X- Coordinate of trend control	R/W
Width	uint	Specifies the width value of trend control	R/W
Height	uint	Specifies the height value of trend control	R/W
CurrentQuality	HmiQuality	Specifies the connection status of trend control	R
Name	string	Specifies the name of trend control	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Enabled	bool	Specifies if allow operator control is enabled or not	R/W
TabIndex	ushort	Screen items specifying a tab index of 0 are not part of the tab order	R/W
ToolBar	HmiToolBarPart	Specifies the tool bar of trend control	R
StatusBar	HmiStatusBarPart	Specifies the status bar on trend control	R
Font	HmiFontPart	Specifies the font type of trend control	R
TimeZone	int	Specifies the time zone on trend control	R/W
ShowRuler	bool	Specifies the ruler of trend control	R/W
ShiftAxes	bool	Specifies the shift axis of trend control	R/W
BackColor	Color	Specifies the background color of trend control	R/W
Legend	HmiLegendPart	Specifies the legend of trend control	R
EventHandlers	HmiTrendControlEventHandlerComposition	Specifies the event handlers of trend control	R
CaptionColor	Color	Specifies the caption color of trend control	R/W
ShowFocusVisual	bool	Specifies if the show focus is enabled or not	R/W

Note

The LabelFont is a Time Axis property of trend base controls.

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access basic properties of trend control modify the following program code:

```
private void TrendControlPropertiesAccess()]
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiScreen screen = hmiSoftware.Screens.Create("Screen_164");
HmiFunctionTrendControl hmitrendcontrol =
screen.ScreenItems.Create<HmiFunctionTrendControl>("CTrendControl");
// To access basic properties
//Width
var autoplay = hmitrendcontrol.Width;
hmitrendcontrol.Width = 1;
//BackColor
var backcolor = hmitrendcontrol.BackColor;
hmitrendcontrol.BackColor = Color.Beige;
//Name
var name = hmitrendcontrol.Name;
hmitrendcontrol.Name = "DefaultName";
// To access multilingual property
//Caption
var caption = hmitrendcontrol.Caption.Items[0].Text;
hmitrendcontrol.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
//To access other typical properties
//Status bar
var statusBar = hmitrendcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
}
```

Accessing trend companion properties (RT Unified)

Property

The following properties are supported in trend companion screen items.

Property Name	Property Type	Description	Accessibility
TrendCompanion-Mode	HmiTrendCompanion-Mode	Specifies the companion mode	R/W
UseSourceControl-TrendColors	bool	Specifies the source control trend color	R/W

Property Name	Property Type	Description	Accessibility
ShowAlways	bool	Specifies if the show always is enabled or not	R/W
UseSourceControl-BackColor	bool	Specifies the source control background color	R/W
Caption	MultilingualProperty	Specifies the text to be shown in the caption of a screen window or windowed control (Label)	R/W
WindowFlags	HmiWindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W
Icon	string	Specifies the icon on the control window	R/W
Top	int	Specifies the value of Y- coordinates of trend companion	R/W
Left	int	Specifies the value of X- Coordinate of trend companion	R/W
Width	uint	Specifies the width value of trend companion	R/W
Height	uint	Specifies the height value of trend companion	R/W
CurrentQuality	HmiQuality	Specifies the connection status of trend companion	R
Name	string	Specifies the name of trend companion	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Enabled	bool	Specifies if allow operator control is enabled or not	R/W
TabIndex	ushort	Screen items specifying a tab index of 0 are not part of the tab order	R/W
ToolBar	HmiToolBarPart	Specifies the tool bar of trend companion	R
StatusBar	HmiStatusBarPart	Specifies the status bar on trend companion	R
SourceTrendControl	string	Specifies the color of trend based on source control	R/W
TimeZone	int	Specifies the time zone of trend companion	R/W
SnapToSourceControl	bool	Specifies the control that should be snapped to	R/W
TrendRulerView	HmiDataGridViewPart	Specifies the trend ruler view	R
TrendStatisticAreaView	HmiDataGridViewPart	Specifies the statistic area view	R
BackColor	Color	Specifies the background color of trend companion	R/W
TrendStatisticResultView	HmiDataGridViewPart	Specifies the statistic result view	R
EventHandlers	HmiTrendCompanionEventHandlerComposition	Specifies the event handlers of trend companion	R
CaptionColor	Color	Specifies the caption color of trend companion	R/W
ShowFocusVisual	bool	Specifies if the show focus is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Pogram code

Modify the following program code to access basic properties of trend companion:

```
private void TrendCompanionPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
var screen = hmiSoftware.Screens;
var createdscreen = screen.Create("TestScreen_1566");
var hmitrendcompanion =
createdscreen.ScreenItems.Create<HmiTrendCompanion>("HmiTrendCompanion_1");
// To access basic properties
//Width
var autoplay = hmitrendcompanion.Width;
hmitrendcompanion.Width = 100;
//BackColor
var backcolor = hmitrendcompanion.BackColor;
hmitrendcompanion.BackColor = Color.Beige;
//Name
var name = hmitrendcompanion.Name;
hmitrendcompanion.Name = "DefaultName";
// To access multilingual property
//Caption
var caption = hmitrendcompanion.Caption.Items[0].Text;
hmitrendcompanion.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></
body>";
// To access other typical properties
//Status bar
var statusBar = hmitrendcompanion.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
}
```

Accessing web control properties (RT Unified)

Property

The following properties are supported in web control screen items.

Property Name	Property Type	Description	Accessibility
Url	string	Specifies the url of the web control	R/W
BackColor	Color	Specifies the background color of web control	R/W

Property Name	Property Type	Description	Accessibility
Caption	MultilingualProperty	Specifies the text to be shown in the caption of a screen window or windowed control (Label)	R/W
WindowFlags	HmiWindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W
Icon	string	Specifies the icon on the control window	R/W
Top	Int	Specifies the value of Y- coordinates of web control	R/W
Left	int	Specifies the value of X- Coordinate of web control	R/W
Width	uint	Specifies the width value of web control	R/W
Height	uint	Specifies the height value of web control	R/W
CurrentQuality	HmiQuality	Specifies the connection status of web control	R/W
Name	string	Specifies the name of web control	R/W
Visible	bool	Specifies if the visible is enabled or not	R/W
Enabled	bool	Specifies if allow operator control is enabled or not	R/W
TabIndex	ushort	Screen items specifying a tab index of 0 are not part of the tab order	R/W
ToolBar	HmiToolBarPart	Specifies the tool bar of web control	R
StatusBar	HmiStatusBarPart	Specifies the status bar on web control	R
EventHandlers	HmiWebControlEventHandlerComposition	Specifies the event handler of web control	R
CaptionColor	Color	Specifies the caption color of web control	R/W
ShowFocusVisual	bool	Specifies if the show focus is enabled or not	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

Modify the following program code to access basic properties of web control:

```
private void WebControlPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
var screen = hmiSoftware.Screens;
var createdscreen = screen.Create("TestScreen_1566");
var hmiwebcontrol = createdscreen.ScreenItems.Create<HmiWebControl>("HmiWebControl_1")
// To access basic property
//Width
var autoplay = hmiwebcontrol.width;
hmiwebcontrol.Width = 100;
//BackColor
var backcolor = hmiwebcontrol.BackColor;
hmiwebcontrol.BackColor = Color.Beige;
//Name
var name = hmiwebcontrol.Name;
hmiwebcontrol.Name = "DefaultName";
// To access multilingual property
//Caption
var caption = hmiwebcontrol.Caption.Items[0].Text;
hmiwebcontrol.Caption.Items[0].Text= "<body><p>TestforMultilingualProperty</p></body>";
// To access other typical properties
//Status bar
var statusBar = hmiwebcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
}
```

Faceplate Container (RT Unified)

Description FaceplateContainer (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on a faceplate container screenitems you can use enumerate, find, index commands and the Contains() method.

To access faceplate container from screen items list by using ScreenItems property of the HmiScreen class modify the following program code:

```
private void FacePlateContainerSearch(HmiSoftware hmiSoftware)
{
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiScreen.Screens[0].ScreenItems;
foreach (var item in screenitems)
{
//working with screenitems
}
}
```

To access faceplate container from screen items list by name modify the following program code:

```
private void FacePlateInstanceSearchByName(HmiSoftware hmiSoftware, string
FacePlateContainerName = "FaceplateContainer_1")
{
HmiScreenItemBase screenitems =
hmiSoftware.Screens[0].ScreenItems.Find(FacePlateContainerName);
}
```

To access faceplate container from screenitems list by index modify the following program code:

```
private void FaceplateContainerSearchByIndex(HmiSoftware hmiSoftware)
{
HmiScreenItemBase screenitem = hmiSoftware.Screens[0].ScreenItems[0];
}
```

To check a particular faceplate container exist in screen item list by using Contains() modify the following program code:

```
private void IsFaceplateContainerExist(HmiSoftware hmiSoftware)
{
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiFaceplateContainer faceplate = screenitems.Create<HmiFaceplateContainer>("Faceplate_1");
bool isexists = screenitems.Contains(faceplate);
}
```

Working with faceplate container

You can perform the following tasks with faceplate container while using TIA Portal Openness:

- Creating faceplate container : FaceplateContainer.Create() (Page 789)
- Deleting faceplate container: FaceplateContainer.Delete() (Page 789)
- Faceplate container properties Accessing faceplate container properties (Page 790)

- Faceplate container contained type property: Accessing contained type properties (Page 793)
- Faceplate container UDT property: Assigning UDT to faceplate container (Page 794)

FaceplateContainer.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a faceplate container screen item modify the following program code:

```
private void CreateFaceplateContainer(HmiSoftware hmiSoftware, string
FacePlateContainerName = "FaceplateContainer_1")
{
HmiScreen hmiScreen= hmiSoftware.Screens.Create(screenName);
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiFaceplateContainer faceplate =
hmiScreen.ScreenItems.Create<HmiFaceplateContainer>(FacePlateContainerName);
}
```

FaceplateContainer.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete faceplate container screen item modify the following program code :

```
private void DeleteFaceplateContainer(HmiSoftware hmiSoftware, string
FaceplateContainerName = "FaceplateContainer_1")
{
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiFaceplateContainer faceplate =
hmiScreen.ScreenItems.Create<HmiFaceplateContainerName>(FaceplateContainerName);
If (faceplate! = null)
{
faceplate.Delete ();
}
}
```

```
private void DeleteFaceplateContainer(HmiSoftware hmiSoftware, string
FacePlateContainerName = "FaceplateContainer_1")
{
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiFaceplateContainer faceplate =
hmiScreen.ScreenItems.Create<HmiFaceplateContainer>(FacePlateContainerName);
IEngineeringObject ObjhmiLineEnggObj = faceplate;
if (ObjhmiLineEnggObj != null)
{
ObjhmiLineEnggObj.Invoke ("Delete", null);
}
}
```

Accessing faceplate container properties (RT Unified)

Property

The following properties are supported in faceplate container:

Property Name	Datatype	Description	Accessibility
Name	System.String	Specifies the name of the faceplate instance	R/W
ContainedType	System.String	Specifies the contained faceplate type For information on contained type, see Accessing contained type properties (Page 793)	R/W
Enabled	System.Boolean	Specifies if the allow operator control is enabled or not	R/W
CurrentQuality	HmiQuality	Specifies the connection status	R
Height	System.UInt32	Specifies the height of control window	R/W
Icon	System.String	Specifies the icon on the control window	R/W
Caption	Siemens.Engineering.MultilingualTex	Specifies the text to be shown in the caption of a screen window or windowed control (Label)	R
TabIndex	System.UInt16	Specifies the screen items specifying a tab index of 0 are not part of the tab order	R/W
Visible	System.Boolean	Specifies the visiblity of screen item	R/W
Width	System.UInt32	Specifies the width of the control window	R/W
WindowFlags	Siemens.Engineering miUnified.UI.Enum WindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W

Property Name	Datatype	Description	Accessibility
Top	System.Int32	Specifies the value of Y coordinates of control window	R/W
Left	System.Int32	Specifies the value of X coordinates of control window	R/W
Interface	Siemens.Engineering.HmiUnified.UI.Parts.HmiFaceplateInterfaceCompostion	Specifies the Faceplate interfaces	R
EventHandlers	HmiFaceplateContainerEventHandlerComposition	Specifies the event handlers of HmiFaceplateContainer For more information on event handler, see	R
ShowFocusVisual	Boolean	Specifies if the show focus is enabled or not	R/W
RotationAngle	Int16	Specifies the rotation angle of the screen item in degree	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To get the properties usage of faceplate container modify the following program code:

```
private void FacePlateContainerPropertiesAccess()
{
HmiScreen screen = m_hmiSoftware.Screens[0];
HmiFaceplateContainer faceplate = screens[0].ScreenItems[0] as HmiFaceplateContainer;
//Get_faceplateContainedType
Var containedtype = faceplate.ContainedType;
Console.Writeline("Faceplate Container Type: " + containedtype);
Example Result = "Default Version No.\Faceplate type name";
//Get_CaptionText
string Caption = faceplate.Caption.Items[0].Text;
Console.WriteLine("Faceplate Caption: " + Caption);
//Get_WindowFlags
HmiWindowFlag WindowFlags = faceplate.WindowFlags;
Console.WriteLine("Faceplate WindowFlags: " + WindowFlags);
//Get_Icon
string Icon = faceplate.Icon;
Console.WriteLine("Faceplate Icon: " + Icon);
//Get_Top
int Top = faceplate.Top;
Console.WriteLine("Faceplate Top: " + Top);
//Get_Left
int Left = faceplate.Left;
Console.WriteLine("Faceplate Left: " + Left);
//Get_Width
uint Width = faceplate.Width;
Console.WriteLine("Faceplate Width: " + Width);
//Get_Height
uint Height = faceplate.Height;
Console.WriteLine("Faceplate Height: " + Height);
//Get_CurrentQuality
HmiQuality CurrentQuality = faceplate.CurrentQuality;
Console.WriteLine("Faceplate CurrentQuality: " + CurrentQuality);
//Get_Name
string Name = faceplate.Name;
Console.WriteLine("Faceplate Name : " + Name);
//Get_Visible
bool Visible = faceplate.Visible;
Console.WriteLine("Faceplate Visible: " + Visible);
//Get_Enabled
bool Enabled = faceplate.Enabled;
Console.WriteLine("Faceplate Enabled: " + Enabled);
//Get_TabIndex
ushort TabIndex = faceplate.TabIndex;
Console.WriteLine("Faceplate TabIndex: " + TabIndex);
}
```

To set the properties usage of faceplate container modify the following program code:

```
private void FacePlateContainerPropertiesAccess()
{
//Set_ContainedType
faceplate.ContainedType = "V0.0.1\Faceplate_1";
//Set_CaptionText:
faceplate.Caption.Items[0].Text = "testCaption";
//Set_WindowFlags
faceplate.WindowFlags = HmiWindowFlag.CanSize;
//Set_Icon
faceplate.Icon = “testCaption”;
//Set_Top
faceplate.Top = 50;
//Set_Left
faceplate.Left = 50;
//Set_Width
faceplate.Width = 50;
//Set_Height
faceplate.Height = 50;
//Set_Name
faceplate.Name = “TestName”;
//Set_Visible
faceplate.Visible = true;
//Set_Enabled
faceplate.Enabled = true;
//Set_TabIndex
faceplate.TabIndex = 10;
}
```

Accessing contained type properties (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the contained type property of faceplate container with new location of faceplate types in project library:

```
private void AccessUnifiedFaceplateProperties(createFacePlateContainer)
{
HmiFaceplateContainer faceplatecontainer = createFacePlateContainer("FacePlateContainer");
faceplateContainer ContainedType = "Faceplate_1";
var containedType = faceplateContainer.ContainedType;
//Output: "V0.0.2\\Faceplate_1
HmiFaceplateContainer faceplatecontainer = CreateFacePlateContainer("FacePlateContainer");
faceplateContainer ContainedType = "V0.0.1\Faceplate_1";
var containedType = faceplateContainer.ContainedType;
//Output: "V0.0.1\\Faceplate_1"
}
```

Assigning UDT to faceplate container (RT Unified)

Requirement

The HMI Software object is accessible
See Description HMISoftware (Page 653)

Program code

To assign UDT to the Faceplate container modify the following program code:

```
private void AssignUDTTToFaceplateContainer(HmiSoftware hmiSoftware, Project proj)
{
  var hmiSoftware = GetHmiSoftware(proj);
  HmiScreenItemBaseComposition screenitems =
hmiSoftware.Screens.Find("Screen_1").ScreenItems;
  HmiFaceplateContainer faceplateContainer = screenitems.Find("FaceplateContainer_1") as
HmiFaceplateContainer;
  faceplateContainer.ContainedType = "Faceplate_1";
  // where UDT is the instance created in TagTable from the UDT created in library
  faceplateContainer.Interface[0].Value = "UDT";
}
```

My controls (RT Unified)

Accessing custom web control container properties (RT Unified)

Property

The following properties are supported in HMI Custom Web Control Container:

Property Name	Property Type	Description	Accessibility
Authorization	string	Specifies the authorization of custom web control container	R/W
EventHandlers	HmiCustomWebControlContainerEventHandlerComposition	Specifies the event handlers of custom web control container	R
Interface	HmiCustomControlInterfaceComposition	Specifies the interface of custom web control container	R
RequireExplicitUnlock	bool	Specifies if the explicit unlock required or not	R/W
Left	int	Specifies the value of the X coordinates of control window	R/W
Caption	HmiTextPart	Specifies the text to be shown in the caption of a screen window or windowed control (Label)	R
CaptionColor	Color	Specifies the caption color of custom web control container	R/W
ContainedType	string	Specifies the contained faceplate type	R/W
CurrentQuality	HmiQuality	Specifies the connection status of custom web control container	R
Dynamization	DynamizationBaseComposition	Specifies the dynamization of custom web control container	R
Enabled	bool	Specifies if allow operator control is enabled or not	R/W
Height	uint	Specifies the height of custom web control container	R/W
Icon	string	Specifies the icon of custom web control container	R/W
Name	string	Specifies the name of custom web control container	R/W
PropertyEventHandlers	PropertyEventHandlerComposition	Specifies the property event handlers of custom web control container	R
ShowFocusVisual	bool	Specifies if the show focus is enabled or not	R/W
TabIndex	ushort	Screen items specifying a tab index of 0 are not part of the tab order	R/W
Top	int	Specifies the value of the Y coordinates of custom web control container	R/W
Visible	bool	Specifies the visiblity of custom web control container	R/W

Property Name	Property Type	Description	Accessibility
Width	uint	Specifies the width of the control window	R/W
WindowFlags	HmiWindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W

Note

The path to access the HMI Custom Web Control Container available in the following location: C:\Program Files\SiemensAutomation\Portal V*\Data\Hmi\CustomControls

Where V* refers to the installed version of TIA Portal.

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access plant overview control of type Hmi custom web control container modify the following program code:

```
private void CustomWebControlContainerPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware(MyProject);
HmiScreenComposition screens = hmiSoftware.Screens;
HmiScreen screen = screens.Create("Screen_1");
HmiCustomWebControlContainer plantOverview =
screen.ScreenItems.Create<HmiCustomWebControlContainer>("DefaultName", "Siemens.CPM");
HmiCustomWebControlContainer reports =
screen.ScreenItems.Create<HmiCustomWebControlContainer>("DefaultName_1", "Siemens.REP");
// Access basic properties of plant overview control of type HmiCustom Web Control Container
//Left
plantOverview.Left = 1;
//CaptionColor
plantOverview.CaptionColor = Color.AliceBlue;
// Access other typical properties of plant overview control of type Hmi Custom Web Control
Container
HmiCustomControlInterfaceComposition customControlInterfaceComposition =
plantOverview.Interface;
var count = customControlInterfaceComposition.Count;
if (count > 0)
{
foreach (var customControlInterface in customControlInterfaceComposition)
{
string propertyName = customControlInterface.PropertyName;
object propertyValue = customControlInterface.Value;
if (propertyName == "SelectionBackColor")
{
customControlInterface.Value = Color.Blue;
}
}
///*--------------------Alternative  Way-------------------------*/
HmiCustomControlInterface selectionBackColorProperty  =
customControlInterfaceComposition.FirstOrDefault(x => x.PropertyName ==
"SelectionBackColor");
if (selectionBackColorProperty != null)
{
selectionBackColorProperty.Value = Color.AliceBlue;
}
}
}
```

Note

To create a custom web control, the unique name mentioned in the manifest.json file of the custom web control should be provided as ContainedTypeValue. Using Display name in this case will not create the custom web control object. As in Example above “Siemens.CPM” is used instead of the display name : Plant overview.

Dynamic widgets (RT Unified)

Accessing custom widget container properties (RT Unified)

Property

The following properties are supported in HMI Custom Widget Container:

Property Name	Property Type	Description	Accessibility
Authorization	string	Specifies the authorization of custom widget container	R/W
CurrentQuality	HmiQuality	Specifies the connection status of custom widget container	R
Dynamizations	DynamizationBaseComposition	Specifies the dynamization of custom widget container	R
Enabled	bool	Specifies if the allow operator control is enabled or not	R/W
EventHandlers	HmiCustomWidgetContainerEventHandlerComposition	Specifies the event handler of custom widget container	R
Height	uint	Specifies the height of custom widget container	R/W
Interface	HmiCustomControlInterfaceComposition	Specifies the interface of custom widget container	R
Left	int	Specifies the value of the X coordinates of custom widget container	R/W
Name	string	Specifies the name of custom widget container	R/W
Opacity	string	Specifies the opacity of custom widget container	R/W
PropertyEventHandlers	PropertyEventHandlerComposition	Specifies the property event handler of custom widget container	R
RequireExplicitUnlock	bool	Specifies if the explicit unlock is required or not	R/W
RotationAngle	short	Specifies the rotational angle of custom widget container	R/W
RotationCenterPlacement	HmiRotationCenterPlacement	Specifies the pivot of custom widget container	R/W
RotationCenterX	float	Specifies the X pivot of custom widget container	R/W
RotationCenterY	float	Specifies th Y pivot of customw widget container	R/W
ShowFocusVisual	bool	Specifies if the show focus is enabled or not	R/W
TabIndex	ushort	Screen items specifying a tab index of 0 are not part of the tab order	R/W
ToolTipText	MultilingualText	Specifies the tooltip text of custom widget container	R

Property Name	Property Type	Description	Accessibility
Top	int	Specifies the value of the Y coordinates of custom widget container	R/W
Visible	bool	Specifies the visiblity of custom widget container	R/W
VisualizeQuality	bool	Specifies the visual quality of custom widget container	R/W
Width	uint	Specifies the width of the control widget container	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access basic properties of "extended.Flame" of type HMI Custom Widget Container modify the following program code:

```
private void CustomWidgetContainerPropertiesAccess()
{
HmiSoftware hmiSoftware = GetHmiSoftware(MyProject);
HmiScreenComposition screens = hmiSoftware.Screens;
HmiScreen screen = hmiSoftware.Screens.Create("Screen_1");
HmiCustomWidgetContainer flame =
screen.ScreenItems.Create<HmiCustomWidgetContainer>("Flame_1", "extended.Flame");
//Left
flame.Left = 1;
//ToolTipText
flame.ToolTipText.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
}
```

Note

Correct value corresponding to parameter 'containedTypeValue' is must, in-order to create custom control.

The keyword "extended.<Widgetname>" should be used for creating any custom widget container using TIA Portal Openness as we use Original name mentioned in the Manifest instead of using a Displayname.

PropertyEventHandlers (RT Unified)

Description PropertyEventHandlers (RT Unified)

Requirement

- The HMI Unified Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access property event of screen items modify the following program code:

```
private void AccessPropertyEvent(HmiSoftware hmiSoftware)
{
  HmiScreen screen = ((HmiSoftware)targetSW).Screens.Create("MyScreen");
  HmiCircle hmiCircle = screen.ScreenItems.Create<HmiCircle>("MYCircle");
  var hmiScreenPropeventDyn = screen.PropertyEventHandlers.Create("Enabled",
PropertyEventType.Change);
  var enabledEvent = screen.PropertyEventHandlers.Find("Enabled",
PropertyEventType.Change);
  //Property Event Browse
  Console.WriteLine("CountofEventActions"+ screen PropertyEventHandlers.Count.ToString());
}
```

Working with property event for screen items

You can perform the following tasks with event for screen/screen items while using TIA Portal Openness:

- Creating Property Event Handlers : PropertyEventHandlers.Create() (Page 800)
- Deleting Property Event Handlers PropertyEventHandlers.Delete() (Page 801)
- Property Event Handlers properties: Accessing PropertyEventHandlers properties (Page 801)

PropertyEventHandlers.Create() (RT Unified)

Requirement

- The HMI Unified Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create property event of screen item level object using PropertyEventHandlers navigator modify the following program code:

```
private void CreatePropertyEventHandler(HmiSoftware hmiSoftware)
{
  HmiScreen screen = ((HmiSoftware)targetSW).Screens.Create("MyScreen");
  HmiCircle hmiCircle = screen.ScreenItems.Create<HmiCircle>("MYCircle");
  //Event Creation
  PropertyEventHandler propertyEvent =
hmiCircle.PropertyEventHandlers.Create("ToolTipText", PropertyEventType.Change);
}
```

PropertyEventHandlers.Delete() (RT Unified)

Requirement

- The HMI Unified Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete property events for screen items modify the following program code:

```
private void DeletePropertyEventHandler()
{
  HmiSoftware hmiSoftware = GetHmiSoftware();
  PropertyEventHandler propertyEvent =
((HmiSoftware)targetSW).Screens[0].ScreenItems[0].PropertyEventHandlers.Create("ToolTipTex
t",   PropertyEventType.Change);
  var changedEvent = screen.PropertyEventHandlers.Find(“ToolTipText”,
PropertyEventType.Change);
  //Property Event Deletion
  changedEvent.Delete();
}
```

Accessing PropertyEventHandlers properties (RT Unified)

Requirement

- The HMI Unified Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access property event properties modify the following program code:

```
private void AccessPropertyEvent()
{
  //Property Event Get
  PropertyEventType eventType = propertyEvent.EventType;
  string propertyName = propertyEvent.PropertyName;
  IHmiScript hmiScript = propertyEvent.Script;
  //Script Action in Event Get
  bool hmiScriptEventAsync = hmiScript.Async;
  string hmiScriptEventGlobal = hmiScript.GlobalDefinitionAreaScriptCode;
  string hmiScriptEventCode = hmiScript.ScriptCode;
  HmiValidationResult result = hmiScript.SyntaxCheck();
}
```

Accessing cross reference service on screenitems (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

```
private void AccessCrossReferenceService()
{
  SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
  HmiSoftware hmiSoftware = softwareContainer.Software as HmiSoftware;  //Accessing lower
levels until we reach the ScreenComposition.
  ScreenComposition screenComposition = hmiSoftware.Screens;
  Screen screen = screenComposition.Find("Screen_1");
  HmiScreenItemBase hmiScreenControl = screen.ScreenItems.Find("Alarm control_1");
  if (hmiscreenControl != null)
  {
  try
  {
  CrossReferenceService crossReferenceService =
hmiScreenControl.GetService<CrossReferenceService>();
  // ......
  }
  catch (Exception e)
  {
  Console.WriteLine(e.Message);
  }
  }
}
```

5.13.1.14 ScreenGroups (RT Unified)

Description ScreenGroups (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access all screen groups of device modify following program code:

```
private void ScreenGroupBrowse(HmiSoftware hmiSoftware)
{
  screenGroups = hmiSoftware.ScreenGroups;
  MessageBox.Show(screenGroups.Count.ToString());
  HmiScreenGroupComposition screenGroup2ndLevel = hmiSoftware.ScreenGroups[0].Groups;
  MessageBox.Show(screenGroup2ndLevel.Count.ToString());
}
```

Working with screen groups

You can perform the following tasks with screen groups while using TIA Portal Openness:

- Creating screen groups:ScreenGroups.Create() (Page 803)
- Deleting screen groups: ScreenGroups.Delete() (Page 804)
- Screen groups properties: Accessing screen groups properties (Page 804)

ScreenGroups.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a screen groups (HmiScreenGroup) with Create method of class HmiScreenGroupComposition modify the following progam code:

```
private void CreateScreenGroupFirstLevel(HmiSoftware hmiSoftware)
{
  HmiScreenGroupComposition screenGroups = hmiSoftware.ScreenGroups;
  HmiScreenGroup screenGroup1stLevel = hmiSoftware.ScreenGroups.Create("Group_1stLevel");
}
```

ScreenGroups.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a specific group with Delete() method of HmiScreenGroup class modify the following program code:

```
private void ScreenGroupDelete(HmiSoftware hmiSoftware)
{
  HmiScreenGroupComposition screenGroups = hmiSoftware.ScreenGroups;
  HmiScreenGroup screenGroupFindLevelOne = hmiSoftware.ScreenGroups.Find("Group_2");
  screenGroupFindLevelOne.Delete();
}
```

Accessing screen groups properties (RT Unified)

Property

The following properties are supported in screen groups:

Property name	Data type	Description	Access
Name	String	Specifies the screen group name	R/W
Groups	HmiScreenGroupComposition	Specifies the list of screen groups	R
Screens	HmiScreenComposition	Specifies the list of screens	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the Groups property of screen groups modify the following program code:

```
private void AccessScreenGroupProperties(HmiSoftware hmiSoftware)
{
  HmiScreenGroupComposition screenGroups = hmiSoftware.ScreenGroups;
  MessageBox.Show(screenGroups.Count.ToString());
  HmiScreenGroupComposition screenGroup2ndLevel = hmiSoftware.ScreenGroups[0].Groups;
  MessageBox.Show(screenGroup2ndLevel.Count.ToString());
}
```

5.13.1.15 Tags (RT Unified)

Description Tags (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on tag you can either use enumerate tag or use Find().

```
private void TagBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all the tags of device
HmiTagComposition hmiTags = hmiSoftware.Tags;
for each (HmiTag hmiTag in hmiTags)
{
//working with tags
}
}
```

```
private void TagSearch()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all the tags of given tag table
string tagTableName = "SpecificTagTable";
HmiTagComposition hmiTags = hmiSoftware.HmiTagTables.Find("tagTableName").Tags;
for each (HmiTag hmiTag in hmiTags)
{
//working with tags
}
}
```

To access a single tag by name modify the following code:

```
private void AccessTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//Search for a specific tag
String hmiTagName = "HMI_Tag_1";
HmiTag hmiTag = hmiSoftware.HmiTags.Find("hmiTagName");
if(hmiTag != null)
{
// working with tags
}
```

To access a tag by table name modify the following code:

```
private void AccessTagByName()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for a tag present in given tag table with name "TagTable1"
HmiTag hmiTag4 = hmiSoftware.HmiTagTables.Find("TagTable1").HmiTags.Find("Tag1");
}
```

Working with tags

You can perform the following tasks with tags while using TIA Portal Openness:

- Creating tags:Tags.Create() (Page 807)
- Deleting tags: Tags.Delete() (Page 808)
- Tag properties: Accessing tag properties (Page 809)
- Tag cross reference service: Accessing cross reference service on tags (Page 816)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

HMISoftware object (Page 651)

Tags.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a HMI tag with Create() of HmiTagComposition class modify the following program code :

```
private void CreateTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition hmiTagscomp1 = hmiSoftware.HmiTags;
//Create a tag in the default tag table
HmiTag hmiTag1 = hmiTagscomp1.Create("Tag1");
//Create a tag in specific tag table with name "TagTable1"
//TagTable1 has to exist in your project otherwise it results in recoverable exception
HmiTag hmiTag2 = hmiTagscomp1.Create("Tag2", "TagTable1");
}
```

To create a HMI tag by accessing Tags property of TagTable object modify the following program code :

```
private void CreateTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition hmiTagcomp2 = hmiSoftware.HmiTagTables.Find("Default tag
table").HmiTags;
//Create a tag in the default tag table.
HmiTag hmiTag3 = hmiTagcomp2.Create("Tag_3");
//Create a tag in specific tag table with name "TagTableName"
//TagTableName has to exist in your project otherwise it results in
recoverable exception
HmiTag hmiTag4 = hmiTagcomp2.Create("Tag_4", "TableTableName");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Tags.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a HMI tag with Delete() of HmiTag class modify the following program code :

```
private void DeleteTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//Delete a tag at HmiSoftware level
HmiTagComposition hmiTagscomp1 = hmiSoftware.HmiTags;
HmiTag hmiTag1 = hmiTagscomp1.Find("Tag1");
hmiTag1.Delete();
}
```

To delete a HMI tag by accessing Tags property of TagTable object modify the following program code:

```
private void DeleteTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//Delete a tag with tag name "Tag2" in tag table
HmiTagComposition hmiTags2 = hmiSoftware.HmiTagTables.Find("Default tag table").HmiTags;
HmiTag hmiTag2 = hmiTags2.Find("Tag2");
hmiTag2.Delete();
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing tag properties (RT Unified)

Property

The following properties are supported in tag, which are also applicable to tag with UDT (user defined data type) and its members.

Property name	Data type	Description	Access
AccessMode	HmiAccessMode	Specifies the access mode of tag	R/W
AcquisitionCycle	String	Speciifies the acquisition cycle attribute of tag	R/W
AcquisitionMode	HmiTagAcquisitionMode	Specifies the acquisition mode of tag	R/W
Comment	MultilingualText	Specifies the get and set comment on tag	R/W
Connection	String	Specifies the connection of tag	R/W
DataType	String	Specifies the data type of tag You can assign different types of datatype for tag for example simple, user defined datatype, and array. To assign user defined datatype to tag, please refer the instruction mentioned in Assigning user defined datatype to tag (Page 812)	R/W
MaxLength	UInt32	Specifies the length of tag	R/W
Address	String	Specifies the address of tag	R/W
InitialMinValue	LowerRange	Specifies the start value for the event of tag	R
Name	String	Specifies the name of tag	R/W
Persistent	Boolean	Specifies the persistence of tag	R/W
PlcName	String	Specifies the name of Plc	R
PlcTag	String	Specifies the Plc Tag attribute	R/W
InitialValue	Object	Specifies the start value of tag	R/W
SubstituteValue	HmiSubstituteValue	Specifies the substitute value of tag	R
UpdateId	UInt32	Specifies the updated id of tag	R/W
InitialMaxValue	UpperRange	Specifies the upper start limit value of tag	R
TagTableName	String	Specifies the tag table name to which tag belongs	R
HmiDataType	String	Specifies the Hmi datatype of tag	R/W

Property name	Data type	Description	Access
LinearScaling	Boolean	Specifies the value range on the Hmi tag and the PLC tag to map each other linearly	R/W
HmiStartValue	Object	Specifies the start value range on the Hmi tag to map linearly to the start value of the PLC tag	R/W
HmiEndValue	Object	Specifies the end value range on the Hmi tag to map linearly to the end value range on the PLC tag	R/W
PlcStartValue	Object	Specifies the start value range on the PLC tag to map linearly to the start value range on the Hmi tag	R/W
PlcEndValue	Object	Specifies the end value range on the PLC tag to map linearly to the end value range on the Hmi tag	R/W
GmpRelevant	Boolean	Specifies the changes to the tag value logged in an audit trail	R/W
TagType	Enum->TagType	Specifies the type of tag for example if tag type is simple, user defined datatype, and array	
Members	HmiTagComposition (HmiTag which is of UDT data type consists of member which itself are Tags.)	Specifies collection of member tags of user defined data type or element tags of array respectively. You can access properties of member tags of user defined datatype, please refer the instruction mentioned in Accessing tag member properties of user defined type (Page 814)	

The following properties are available in LowerRange/UpperRange:

Property Name	Data Type	Description	Access
Value	Object	Specifies the lower and upper value of tag	R/W
ValueType	HmiLimitValueType	Specifies the get and set value type of tag	R/W

The following properties are available in SubstituteValue:

Property Name	Data Type	Description	Access
Value	Object	Specifies the get and set value type	R/W
SubstituteValueUsage	Hmi SubstituteValueUsage	Specifies when to use substitute value	R/W

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of tag independent of data type modify the following program code:

```
private void TagProperties()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//Tag_1 has to exist in TIA device project.
HmiTag hmiTag = hmiSoftware.HmiTags.Find("Tag_1");
//Enumeration type properties
AccessMode accessMode = hmiTag.AccessMode;
UpdateScope scope = hmiTag.UpdateScope;
//Assign valid value to properties which are independent from datatype of the tag
hmiTag.Name = "Tag_2";
hmiTag.AcquisitionCycle = "T1s";
hmiTag.DataType = "int";
hmiTag.PlcTag = "PlcTag_1";
}
```

To access the properties of tag dependent of data type modify the following program code:

```
private void TagProperties()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//Tag_1 has to exist in TIA device project.
HmiTag hmiTag = hmiSoftware.HmiTags.Find("Tag_1");
//Enumeration type properties
AccessMode accessMode = hmiTag.AccessMode;
UpdateScope scope = hmiTag.UpdateScope;
//Substitute value depends on data type of tag.
hmiTag.SubstituteValue.Value = "144";
hmiTag.SubstituteValue.OnCommunicationError = false;
//Start value depends on data type of tag.
hmiTag.StartValue = "1";
hmiTag.LowerLimit.ValueType = LimitValueType.Constant;
hmiTag.LowerLimit.Value = "HmiTag_1";
}
```

To assign a string for a multilingual text then add a translation in every language of project modify the following program code:

```
private void TagProperties()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//Tag_1 has to exist in TIA device project.
HmiTag hmiTag = hmiSoftware.HmiTags.Find("Tag_1");
//Enumeration type properties
AccessMode accessMode = hmiTag.AccessMode;
UpdateScope scope = hmiTag.UpdateScope;
//Comment property (multilingual text)
string culture = "en-US";
//get the language based on given culture. Culture is standard and it should have correct
value.
Language language = GetDeviceProject().LanguageSettings.Languages.Find(new
System.Globalization.CultureInfo(culture));
//get comment as multilingual text.
MultilingualText multiLingualComment = hmiTag.Comment;
//get all text items from comment property.
MultilingualTextItemComposition texts = multiLingualComment.Items;
//find text from text list based on language.
MultilingualTextItem textItemEnglish = texts.Find(language);
//get value in selected culture.
string commentEng = textItemEnglish.Text;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Assigning user defined datatype to tag (RT Unified)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To assign user defined datatype as datatype to tag without connection modify the following program code:

```
private void AssignUdtWithoutConnection()
{
//Assigning three types of UDT to tag by calling DataType property of tag
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].DataType = @"\Project library\Types\HmiUdt_Int\V 0.0.1";
tags[1].DataType = @"\Project library\Types\HmiUdt_byte\V 0.0.1";
tags[2].DataType = @"\Project library\Types\New folder_2\HmiUdt_string\V
0.0.1";
tags[3].DataType = @"\Project library\Types\New folder_1\HmiUdt_string\V
0.0.1";
tags[4].DataType = @"\Project library\Types\New folder_3\HmiUdt_string\V
0.0.1";
}
```

To assign user defined datatype as datatype to tag with connection modify the following program code:

```
private void AssignUdtWithConnection()
{
//Assigning user defined datatype to tag by calling DataType property of
tag with connection
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].Connection = "Connection_1";
tags[0].DataType = @"\Project library\Types\New folder_3\HmiUdt_string\V
0.0.1";
}
```

Note

To assign user defined datatype to tag with connection you should have the correct PLC type connection exist in TIA Portal project; otherwise, a recoverable exception is thrown.

To assign PLC user defined datatype as datatype to tag with connection modify the following program code:

```
private void AssignPlcUdtWithConnection()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].Connection = "HMI_Connection_1";
tags[0].Data Type = "PlcTag_1";
}
```

Note

To assign PLC userdefined datatype as datatype to tag you should have PLC and HMI devices exist in the same TIA Portal project and have integrated connection exist between them.

The PLC device also needs to have a PLC tag with a UDT type.

See also

Opening a project (Page 128)

Accessing tag member properties of user defined type (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access member properties of user defined datatype for tag modify the following program code:

```
private static void GetProperties(HmiTagComposition hmiTags)
{
foreach (HmiTag hmiTag in hmiTags)
{
switch(hmiTag.TagType)
{
case HmiTagType.Simple
Console.WriteLine("Name : "+ hmiTag.Name);
GetSimpleTagProperties(HmiTag hmiTag)
break;
case HmiTagType.UDT;
case HmiTagType.Array;
GetProperties(hmiTag.Members); //recursive call to current function.
break;
}
}
}
//The GetSimpleTagProperties()uses hmiTag object as a parameter to access the properties of
tag.
private static void GetSimpleTagProperties(HmiTag hmiTag)
{
Console.WriteLine("Name : " +hmiTag.Name);
Console.WriteLine("Date type : " +hmiTag.DataType);
// List the properties of hmi tag
//
//...
}
//The GetProperties()uses hmiTag.Members object as a parameter to access the properties of
tag members.
private static void GetProperties(HmiTag hmiTag.Members)
{
Console.Writeline("Name : " +hmiTag.Members.Name);
Console.Writeline("Date type: " +hmiTag.Members.DataType);
// List the properties of hmi tag members
//...
//...
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing cross reference service on tags (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the cross reference service through GetService() on tag object modify the following program code:

```
private void getCrossReferenceServiceforTag()
{
SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
HmiSoftware hmiSoftware = softwareContainer.Software as HmiSoftware;  //Accessing lower
levels until we reach the HmiTagComposition.
HmiTagComposition hmiTagComposition = hmiSoftware.HmiTags;
HmiTag hmiTag = hmiTagComposition.Find("Tag_1");
if (hmiTag != null)
{
try
{
CrossReferenceService crossReferenceService = hmiTag.GetService<CrossReferenceService>();
......
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

HMISoftware object (Page 651)

5.13.1.16 TagTables (RT Unified)

Description TagTables (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To perform a task on a tag table you can either use enumerate tag tables or use Find():

```
private void TagTableBrowse()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can navigate all the tag tables of device
HmiTagTableComposition tagTables = hmiSoftware.TagTables;
foreach (HmiTagTable tagTable in tagTables)
{
//Working with tag table
}
}
```

To access a single tag table by name modify the following program code:

```
private void TagTableSearch()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// Search for a specific tag table name from a list of tag tables
string tagTableName = "TagTable_1"
HmiTagTableComposition tagTables = hmiSoftware.TagTables;
HmiTagTable tagTableObj = tagTables.Find("tagTableName");
//Working with tag table
}
```

Working with tag tables

You can perform the following tasks with tag tables while using TIA Portal Openness:

- Creating tag tables:TagTables.Create() (Page 818))
- Deleting tag tables:TagTables.Delete() (Page 818)
- Exporting tags: TagTable.Tags.Export() (Page 824)
- Importing tags: TagTable.Tags.Import() (Page 824)
- Tag tables properties: Accessing tag table properties (Page 819)

Note

Querying errors is not supported for tag tables as they don't have any properties which can be in error state.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

TagTables.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a tag table with Create() of HmiTagTableComposition class modify the following program code

```
private void TagTableCreate()
{
//Create tag table with name "TagTable_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.TagTables;
HmiTagTable hmiTagTable = tagTables.Create("TagTable_1");
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

TagTables.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a tag table with Delete() of HmiTagTable class modify the following program code:

```
private void TagTableDelete()
{
//Delete tag table with name "TagTable_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.TagTables;
HmiTagTable tagTable = tagTables.Find("TagTable_1");
tagTable.Delete();
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing tag table properties (RT Unified)

Property

The following properties are supported in tag tables:

Property name	Data type	Description	Access
Name	String	Specifies the name of tag table	R/W
Tags	HmiTagComposition	Specifies the list of tags	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the properties of a tag table modify the following program code :

```
private void TagTableProperties()
{
//Set and Get the "Name" properties of tag table.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.TagTables;
HmiTagTable tagTable = tagTables.Find("TagTable_1");
tagTable.Name = "NewTagTable";
string name = tagTable.Name;
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Exporting and importing tag tables (RT Unified)

Introduction

You can use TIA Portal Openness to export and import data for a tag table. In case of import new objects are created under unified device if necessary. The export file is a text file using a simplified format: WinCC ML, which is based on YAML format. You can modify the file according to your requirements.

In TIA Portal Openness the export and Import method are implemented at the tag table level.

Note

It is recommended that you should rectify all compilation errors related to tags in tag tables which shall be exported.

Export of tags

Using Export functionality, the following types of tags will be exported:

- Tags
  - Internal tags
  - External tags
- Structure tags with their elements
  - HMI user data type (UDT)
  - PLC user defined datatype (UDT)
- Array Tags with their elements
  - Internal array tags
  - External array Tags

Note

When an array of WChar and Char datatype is exported then no tag elements are displayed in the exported WinCCML file and the tags are displayed in the simple tags category. When these tags are imported back, they are added as array tags as expected.

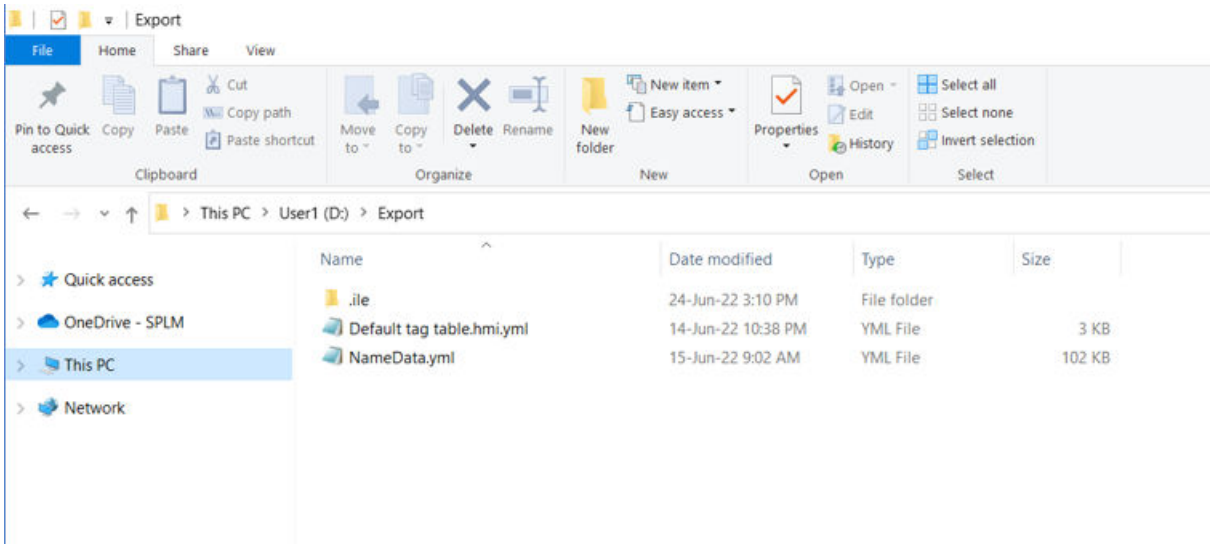
Note

When a structure tag is exported the reference name of the PLC structure tag type will be appended to the PLC name.

Contents of Export location

When a tag table is exported to a particular folder location, the following contents will be created:

- Tag table file with the extension ".hmi.yml" and user defined name in the example "Default tag table.hmi.yml".
- A file "NameData.yml" This file does not contain any data required for executing the import and export
- A folder ".ile" This folder does not contain any data required for executing the import and export
- A file with the extension "def.hmi.yml" will be created when a tag table with structure tags is exported. This file will contain the structure types definition



YAML File structure

The structure of the YAML file will be as seen below. It can be opened & edited in any simple text editor like Notepad, Microsoft word, Notepad++, etc. :

The overall configuration will be seen in property name & property value format which makes it easy to read and modify the configuration.

```
#Version: 2.0
SimpleTags:
Tag_1:
AccessMode: SymbolicAccess
AcquisitionCycle: T1s
Address: 0001:TS:50.73A8C75D.C
Connection: HMI_Connection_2
ControllerDataType: 1006
DataType: Int
InitialMaxValue: 444
InitialMinValue: 44
MaxLength: 2
PlcTag: PLCTag_1
Tag_2:
AccessMode: AbsoluteAccess
AcquisitionCycle: T1s
Address: 0001:CL:%IW0
Connection: HMI_Connection_1
ControllerDataType: 1006
DataType: Int
MaxLength: 2
PlcTag: PLCTag
SymbolicAddress: '%IW0'
Tag_3:
AccessMode: SymbolicAccess
AcquisitionCycle: T1s
Address: $0001:TS:8A0E0001.CDB29A10.A.#{0}
Connection: HMI_Connection_1
ControllerDataType: 1009
DataType: DInt
MaxLength: 4
PlcTag: Data_block_1.Element1[0]
SymbolicAddress: Data_block_1.Element1[Tag_1]
Tag_4
AccessMode: SymbolicAccess
AcquisitionCycle: T1s
Address: 0001:TS:8A0E0001.CDB29A10.A.1
Connection: HMI_Connection_1
ControllerDataType: 1009
DataType: DInt
MaxLength: 4
PlcTag: Data_block_1.Element1[1]
```

Property

The below table shows the mapping of properties name in TIA Portal with the properties name being exported in WinCCML file:

TIA Portal Property Name	WinCCML Property Name	Remark
Name	Heading of object	
DataType	ControllerDataType	For connections other than Internal, this property will be used during import, to assign datatype to the tag.
Connection	Connection	
PLC Name	Derived from PLC Tag property	
PLC Tag	PlcTag	
Address	SymbolicAddress	Property named as 'Address' is ignored by TIA Portal ES during Import
Access Mode	AccessMode	
HMI data type	DataType	For Internal connection Tags, this property will be used during import, to assign datatype to the tag.
Length	MaxLength	
Acquisition cycle	AcquisitionCycle	
Acquisition mode	AcquisitionMode	
Upper 2 (Tag)	MaxValueTag	
Lower 2 (Tag)	MinValueTag	
Upper 2 (Constant)	InitialMaxValue	
Lower 2 (Constant)	InitialMinValue	
Linear scaling	Derived from PLC/Hmi Start/End Values	This is set as True if PLC/Hmi Start/End Values are present
End value PLC	PLCMaxValue	
Start value PLC	PLCMinValue	
End value HMI	HMIMaxValue	
Start value HMI	HMIMinValue	
Start value	InitialValue	
Update ID	ID	
Persistent	Persistent	
Substitute value	SubstituteValue	
Use Substitute Value	SubstituteValueUsage	
GMP relevant	GMPRelevant	
Type of confirmation	ConfirmationType	
Comment required	Derived from ConfirmationType value	
Scope	Scope	
Comment	Comment	
	Address	
	Offset	
	Slope	

TagTable.Tags.Export() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To export a tag table modify the following program code:

```
DirectoryInfo directoryInfo = new DirectoryInfo("D:\\Data Exchange");
HmiTagTableComposition tables = GetHmiSoftware(currentProject).TagTables;
HmiTagTable tagTable = tables.Find(tagTableName.Text);
\\Create a file name "ExportedTable" in specific directory by using Export() that includes
the details of tag table.
IList<FileInfo> Exportresult =
tagTable.Tags.Export(directoryInfo,"ExportedTable").ToList();
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

TagTable.Tags.Import() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To import a tag tables modify the following program code:

```
DirectoryInfo directoryInfo = new DirectoryInfo("D:\\Data Exchange");
HmiSoftware ImportHMISoftware = GetHmiSoftware(currentProject);
HmiTagTableComposition tables = ImportHMISoftware.TagTables;
HmiTagTable tagTable = tables.Find(importedTagTable.Text);
bool ImportResult =  tagTable.Tags.Import(directoryInfo);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.1.17 TagTableGroups (RT Unified)

Description TagTableGroups (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access a tag table group using the Find() method modify the following program code:

```
privat void SearchTagTableGroups(HmiSoftware hmiSoftware)
{
  HmiTagTableGroupComposition tagTableGroups = hmiSoftware.TagTableGroups;
  HmiTagTableGroup tagTableGroupFindLevelOne = hmiSoftware.tagTableGroups.Find("Group_2");
  MessageBox.Show("Found at 1st level: " + tagTableGroupFindLevelOne.Name);
}
```

Working with tag table groups

You can perform the following tasks with tag groups while using TIA Portal Openness:

- Creating tag table groups:TagTableGroups.Create() (Page 825)
- Deleting tag table groups: TagTableGroups.Delete() (Page 826)
- Tag table groups properties: Accessing tag table groups properties (Page 826)

TagTableGroups.Create() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To create a tag table group (HmiTagTableGroup) with Create() method of class HmiTagTableGroupComposition modify the following program code:

```
private void CraeteTagTableGroup(HmiSoftware hmiSoftware)
{
  HmiTagTableGroupComposition tagTableGroups = hmiSoftware.TagTableGroups;
  int count = tagTableGroups.Count;
  HmiTagTableGroup tagTableGroup1stLevel =
hmiSoftware.TagTableGroups.Create("Group_1stLevel");
}
```

TagTableGroups.Delete() (RT Unified)

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To delete a tag table group modify the following program code:

```
private void DeleteTagTableGroup(HmiSoftware hmiSoftware)
{
  HmiTagTableGroupComposition tagTableGroups = hmiSoftware.TagTableGroups;
  HmiTagTableGroup tagTableGroupFindLevelOne = hmiSoftware.TagTableGroups.Find("Group_2");
  MessageBox.Show("Found at 1st level: " + tagTableGroupFindLevelOne.Name);
  tagTableGroupFindLevelOne.Delete();
}
```

Accessing tag table groups properties (RT Unified)

Property

The following properties are supported in tag table group:

Property name	Data type	Description	Access
Name	String	Specifies the tag table group name	R/W
Groups	HmiTagTableGroupComposition	Specifies the list of tag table groups	R
TagTables	HmiTagTableComposition	Specifies the list of tag tables	R

Requirement

- The HMI Software object is accessible
  See Description HMISoftware (Page 653)

Program code

To access the Groups property of tag table groups of a device modify the following program code:

```
private void AccessTagTableGroupProperties(HmiSoftware hmiSoftware)
{
  HmiTagTableGroupComposition tagTableGroups = hmiSoftware.TagTableGroups;
  MessageBox.Show(tagTableGroups.Count.ToString());
  HmiTagTableGroupComposition tagTableGroup2ndLevel = hmiSoftware.TagTableGroups[0].Groups;
  MessageBox.Show(tagTableGroup2ndLevel.Count.ToString());
}
```

5.13.2 Accessing further objects (RT Unified)

5.13.2.1 Rename properties and data types (RT Unified)

Introduction

In ES TIA Portal Openness, appropriate changes are made in property names and data types, so that it is similar to runtime Openness model.

To achieve it, changes are made in all subsystems for example Alarm, Tag, Connection, and Logging.

To have consistency in naming the property and its data type, it is necessary that prefix Hmi is used for defining their type and name of property should not have this prefix.

Change in property name and data type

Following table shows required changes to have consistent name and data type for composition.

Hmi Object Name	Property Name	Property New Name	Property Data type	Property New Data Type
Analog Alarm	AnalogAlarms	-	AnalogAlarmComposition	HmiAnalogAlarmComposition
Discrete Alarm	DiscreteAlarms	-	DiscreteAlarmComposition	HmiDiscreteAlarmComposition
OPC UA Alarm & Condition types	OpcUaAlarmTypes	-	OpcUaAlarmTypeComposition	HmiOpcUaAlarmTypeComposition
Alarm Class	AlarmClasses	-	AlarmClassComposition	HmiAlarmClassComposition

Hmi Object Name	Property Name	Property New Name	Property Data type	Property New Data Type
Tag Table	HmiTagTables	TagTables	HmiTagTableComposition	-
Tag	HmiTags	Tags	HmiTagComposition	-
System Tag	HmiSystemTags	SystemTags	HmiSystemTagComposition	-
Logging Tag	LoggingTags	-	LoggingTagComposition	HmiLoggingTagComposition
Connection	HmiConnections	Connections	HmiConnectionComposition	-
Data Log	DataLogs	-	DataLogComposition	HmiDataLogComposition
Alarm Log	AlarmLogs	-	AlarmLogComposition	HmiAlarmLogComposition
Plant Object Interface	PlantObjectTags	-	PlantObjectInterfaceComposition	-
Screen	Screens	-	HmiScreenComposition	-

Note

With dash (-) character in columns 'New Name' and 'New Data Type' in above table shows no change is required.

Changes in name of HmiObject classes

Following table shows required changes to have consistent name for classes for Hmi Object.

Hmi Object Name	Class Name	New Class Name
Analog Alarm	AnalogAlarm	HmiAnalogAlarm
Discrete Alarm	DiscreteAlarm	HmiDiscreteAlarm
OPC UA Alarm & Condition types	HmiOpcUaAlarmType	HmiOpcUaAlarmType
Alarm Class	AlarmClass	HmiAlarmClass
Tag Table	HmiTagTable	-
Tag	HmiTag	-
Connection	HmiConnection	-
RuntimeSettings	RuntimeSetting	HmiRuntimeSetting
Data Log	DataLog	HmiDataLog
Alarm Log	AlarmLog	HmiAlarmLog
Logging Tag	LoggingTag	HmiLoggingTag
System Tag	HmiSystemTag	-
Screen	HmiScreen	-
Plant Object interface	PlantObjectInterface	-

Note

With dash (-) character in columns 'New Class Name' in above table shows no change is required.

5.13.2.2 Querying errors (RT Unified)

Introduction

You can use TIA Portal Openness to get error information about properties for the following WinCC objects:

- Analog Alarm
- Discrete Alarm
- Alarm Class
- Data Log
- Alarm Log
- Tag
- Connection
- Logging Tag
- Device runtime settings
- OPC UA Alarms

Note

This feature is not applicable to System Tag and Tag Table as they don’t have any property which can be in error state.

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128).
- Access to HMI Unified Software object
  See HMI Unified Software object (Page 653)

Program code

Modify the following program code to query errors for tag, alarms, runtime setting and screens:

```
//For Tag
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition hmiTags = hmiSoftware.HmiTags;
HmiTag hmiTag = hmiTags.Find("Tag-1");
if(hmiTag != null)
{
//validate method usage
IList<HmiValidationResult> validationResult = hmiTag.Validate();
if (validationResult = null && validationResult.Count > 0)
{
foreach (HmiValidationResult propertyErrors in validationResult)
{
//browse error for different property
string propertyName = propertyErrors. PropertyName;
foreach (string propertyError in propertyErrors.Errors)
{
//work with propertyerrors
}
}
}
}
//For Discrete Alarm
HmiSoftware hmiSoftware = ...;
HmiDiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
HmiDiscreteAlarm alarm = discreteAlarms.Find("Alarm_1");
if (alarm != null)
{
//Validate Method Usage
Ilist<HmiValidationResult> validationResult = alarm.Validate();
if (validationResult != null && validationResult.Count > 0)
{
foreach (HmiValidationResult propertyErrors in validationResult)
{
//browse error for different property
string propertyName = propertyErrors.PropertyName;
foreach (string propertyError in propertyErrors.Errors)
{
//work with property errors
}
}
}
}
//For Runtime Setting
HmiSoftware hmiSoftware = ...;
HmiRuntimeSetting runtimeSettings = hmiSoftware.RuntimeSettings;
//Validate Method Usage
Ilist<HmiValidationResult> validationResult = runtimeSettings.Validate();
if (validationResult != null && validationResult.Count > 0)
{
foreach (HmiValidationResult propertyErrors in validationResult)
{
//browse error for different property
string propertyName = propertyErrors.PropertyName;
foreach (string propertyError in propertyErrors.Errors)
{
```

```
//work with property errors
}
}
}
// List of Many objects (Tag, Alarm, etc,)
HmiSoftware hmiSoftware = ...;
IList<IValidator> objectsToValidate = new List<IValidator>();
//Put tags in validation list
HmiTagComposition hmiTags = hmiSoftware.Tags;
foreach (HmiTag hmiTag in hmiTags)
{
objectsToValidate.Add(hmiTag);
}
//Put alarms in validation list
HmiDiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
foreach (HmiDiscreteAlarm discreteAlarm in discreteAlarms)
{
objectsToValidate.Add(discreteAlarm);
}
//Put RuntimeSettings in validation list
objectsToValidate.Add(hmiSoftware.RuntimeSettings);
foreach (IValidator validator in objectsToValidate)
Ilist<HmiValidationResult> validationResult = validator.Validate();
if (validationResult != null && validationResult.Count > 0)
{
foreach (HmiValidationResult propertyErrors in validationResult)
{
//browse error for different property
string propertyName = propertyErrors.PropertyName;
foreach (string propertyError in propertyErrors.Errors)
{
//work with property errors
}
}
}
}
```

See also

Description HMISoftware (Page 653)

5.13.2.3 Accessing simple dynamic (RT Unified)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- Access to HMI Software object
  See HMI Unified Software object (Page 653)

Introduction

You can use the TIA Portal Openness to have complete access to the dynamization by simple dynamics. The Screen & its Screen items can be dynamized using one of the following simple dynamization options :

- Range
- Multiple bits
- Single bit

Program code

```
var hmiSoftware = GetScadaHmiSoftware();
var Screen1 = hmiSoftware.Screens[0];
//Range :
var screenItem0 = Screen1.ScreenItems.FirstOrDefault(x => x.Name == "Circle_1");
string propertyName = "BackColor";
var tagDynamization = screenItem0.Dynamizations.Where(x => x.PropertyName ==
propertyName).FirstOrDefault() as TagDynamization;
var mappingTable = tagDynamization.ValueConverter.MappingTable;
var entries = mappingTable.Entries;
mappingTable.ConditionType = ConditionType.Range;
//First Row Entry :
var rangeEntry1 = entries.Create<MappingTableEntryRange>(); //Create a new row
rangeEntry1.From = 0; // Add a From entry
rangeEntry1.To = 50; // Add a To Entry
rangeEntry1.Value = Color.Red; // Set the property value
rangeEntry1.Flashing = true; // Select the Flashing state
rangeEntry1.FlashingRate = FlashingRate.Medium; // Select the Flashing rate
rangeEntry1.AlternateValue = Color.Orange; // Select the Flashing color
//Second Row Entry :
var rangeEntry2 = entries.Create<MappingTableEntryRange>(); //Create a new row
rangeEntry2.From = 51; // Add a From entry
rangeEntry2.To = 80; // Add a To Entry
rangeEntry2.Value = Color.Yellow; // Set the property value
rangeEntry2.Flashing = true; // Select the Flashing state
rangeEntry2.FlashingRate = FlashingRate.Medium; // Select the Flashing rate
rangeEntry2.AlternateValue = Color.YellowGreen; // Select the Flashing color
//Third Row Entry :
var rangeEntry3 = entries.Create<MappingTableEntryRange>(); //Create a new row
rangeEntry3.From = 81; // Add a From entry
rangeEntry3.To = 100; // Add a To Entry
rangeEntry3.Value = Color.Green; // Set the property value
rangeEntry3.Flashing = false; // Select the Flashing state
//Fourth Row Entry :
var rangeEntry4 = entries.Create<MappingTableEntryRange>(); //Create a new row
rangeEntry4.From = 101; // Add a From entry
rangeEntry4.To = 101; // Add a To Entry
rangeEntry4.Value = Color.Green; // Set the property value
rangeEntry4.Flashing = true; // Select the Flashing state
rangeEntry4.FlashingRate = FlasingRate.Medium;
rangeEntry4.AlternateValue = Color.LightGreen;
//Second Row Entry - Bit number 4 :
var multibitEntries2 = entries.Create(BitDynamizationType.MultiBit);
var multibitEntry2 = multibitEntries2[0];
ulong multibitCondition2 = 16; // Condition = 2^(Bit No.) = 2^4 => 16
ulong multibitRelevant2 = 20; // Relevant = Decimal equivalent of no. of bits being
configured = 0001 0100 => 20
multibitEntry2.Update(multibitCondition2, multibitRelevant2);
multibitEntry2.Value = Color.Green;
multibitEntry2.Flashing = false;// Single Bit
var hmiSoftware = GetScadaHmiSoftware();
var Screen1 = hmiSoftware.Screens[0];
var screenItem0 = Screen1.ScreenItems.FirstOrDefault(x => x.Name == "Process control_1");
string propertyName = "Visible";
var tagDynamization = screenItem0.Dynamizations.Where(x => x.PropertyName ==
propertyName).FirstOrDefault() as TagDynamization;
```

```
var mappingTable = tagDynamization.ValueConverter.MappingTable;
var entries = mappingTable.Entries;
var singleBitEntries = entries.Create(BitDynamizationType.SingleBit);
mappingTable.ConditionType = ConditionType.Singlebit; // Select Condition type
//State 0 for Bit 15 :
var singleBitEntry1 = singleBitEntries[0]; // Select the 1st Entry - Off State of the
bitulong singlebitCondition1 = 0; // Condition value = [Bit State x 2^ Bit number] => (0 x
2^15 = 0)ulong singlebitRelevant1 = 32768; // Relevant = Decimal equivalent of the Single
bit being configured = 1000 0000 0000 0000 => 32768
singleBitEntry1.Update(singlebitCondition1, singlebitRelevant1);
singleBitEntry1.Value = true;//State 1 for Bit 15 :
var singleBitEntry2 = singleBitEntries[1]; // Select the 2nd Entry - ON State of the bit
ulong singlebitCondition2 = 32768; // Condition value = [Bit State x 2^ Bit number] => (1 x
2^15 = 32768)
ulong singlebitRelevant2 = 32768; // Relevant = Decimal equivalent of the Single bit being
configured = 1000 0000 0000 0000 => 32768
singleBitEntry2.Update(singlebitCondition2, singlebitRelevant2);
singleBitEntry2.Value = false;
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Description HMISoftware (Page 653)

5.13.2.4 Accessing device name of HMI Panel (RT Unified)

Introduction

You can use the TIA Portal Openness application to read and write the device names of the unified and non-unified HMI device.

Properties

The following attribute can be accessed and edited in HMI device:

Attribute	Data type	Description	Access
Name	String	Name of HMI device	R/W

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82).
- A project is open.
  See Opening a project (Page 128)

Program code

```
Tiaproject = tiaPortal.Projects[0];
var deviceUnderTest = Tiaproject.Devices[0];
//Set
deviceUnderTest.Name = "Test1";
//Get
var deviceName = deviceUnderTest.Name;
foreach (var deviceItem in deviceUnderTest.DeviceItems)
{
//Set
deviceItem.Name = "TestDeviceItem1";
//Get
var deviceName = deviceItem.Name;
}
```

5.13.2.5 Accessing common plant model hierarchies (RT Unified)

Working with plant view (RT Unified)

Introduction

You can perform the following tasks related to plant view while using TIA Portal Openness:

- Create plant view
- Delete plant view
- Rename plant view

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code to create a plant view:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
```

Note

You can create only one plant view inside a project.

Modify the following program code to delete or remove a plant view:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
plantView.Delete();
```

Modify the following program code to rename a plant view by updating the 'Name' property:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
plantView.Name = "New_PlantView_Name";
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Working with plant view nodes (RT Unified)

Introduction

You can perform the following tasks related to plant view node while using TIA Portal Openness:

- Create plant view nodes
- Delete plant view nodes
- Rename plant view nodes

Properties

The following properties are supported in Plant View Node using TIA Portal Openness:

Property Name	Data Type	Accessibility
HierarchyPath	String	R
PlantObject	Siemens.Engineering.HmiUnified.Cpm.PlantObject	R
Name	String	R/W
PlantView	String	R
ViewPath	String	R
PlantViewNodes	PlantViewNodeComposition	R

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code to create plant view node under a plant view:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;
PlantViewNode mixingNode = viewNodes.Create("Mixing");
```

Modify the following program code to create plant view node under another plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;
PlantViewNode mixingNode = viewNodes.Create("Mixing");
PlantViewNode motorNode = mixingNode.Create("Motor");
```

Note

You can create any number of plant view nodes inside a plant view or plant view node itself.

Modify the following program code to remove or delete a plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");
mixingNode.Delete();
```

Modify the following program code to rename a plant view node by updating the 'Name' property:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;
PlantViewNode mixingNode = viewNodes.Create("Mixing");
mixingNode.Name = "New_PlantViewNode_Name";
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Enumerating plant view (RT Unified)

Introduction

You can use the TIA Portal Openness to enumerate plant view present on given project in TIA Portal.

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code to enumerate all plant views of the project:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
foreach (var item in plantViews)
{
// work with plant views (currently only 1 plant view can be created inside a project)
}
```

Modify the following program code to find a plant view from plant view list based on the name:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
```

Modify the following program code to find a plant view from plant view list based on index:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
```

Modify the following program code to find plant view node from the plant view list based on the hierarchy path of the required plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantViewNode plantViewNode =
plantViews.FindPlantViewNode("ABCManufacturingPlan\Mixing\Motor");
```

Modify the following program code to find a plant view node from the plant view node list based on the name of the plant view:

```
PlantViewNodeComposition plantViewNodes =
m_tiaPortalApp.Projects[0].PlantViews[0].PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodes.Find("Mixing");
```

Modify the following program code to enumerate all plant view nodes list available inside plant view:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantViewNodeComposition nodes = plantView.PlantViewNodes;
foreach (var item in nodes)
{
// work with plant view nodes inside plant view
}
```

Modify the following program code to enumerate all plant view nodes available inside any hierarchial level:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantViewNodeComposition nodes = plantView.PlantViewNodes;
PlantViewNodeComposition subNodes = nodes[2].PlantViewNodes;
foreach (var item in subNodes)
{
// work with plant view nodes inside 2nd node within the plant view
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Working with plant view to device (RT Unified)

Introduction

You can perform the following tasks related to accessing plant view hierarchies and plant object while using TIA Portal Openness:

- Assign device to the plant view
- Change device to the plant view
- Remove device to the plant view

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code to assign a device to the plant view by setting the AssignedHmiDevice property:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
plantView.AssignedHmiDevice = "HMI_RT_1";
```

Modify the follwing program code to change a device to the plant view by setting a device name to AssignedHmiDevice property.

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
plantView.AssignedHmiDevice = "HMI_RT_2";
```

Modify the following program code to remove device to the plant view by setting a blank device name to AssignedHmiDevice property.

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
plantView.AssignedHmiDevice = string.Empty;
```

Note

To remove the device, use only string.empty. Passing NULL or whitespace (" "), will be treated as a device name and will return device not found error message

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.2.6 Accessing common plant model instance (RT Unified)

Working with CPM object instance (RT Unified)

Introduction

You can perform the following tasks related to access CPM instance based on CPM type while using TIA Portal Openness:

- Create CPM object instances
- Delete CPM object instances
- Rename CPM object instances

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- Create a Plant

Program code

Modify the following program code to create CPM object instance:

```
// Instance inside plant view
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
plantView.AssignedHmiDevice = "HMI_RT_1";
PlantViewNodeComposition nodes = plantView.PlantViewNodes;
PlantViewNode node = nodes.Create("ObjectLevel1", "Plant_Object_Type_1");
//Instance inside plant view node
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer",
"Plant_Object_Type_1");
```

Modify the following program code to remove or delete a CPM object instance:

```
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Find("IngredientMixer");
mixerObject.Delete();
```

Modify the following program code to rename a CPM object instance by updating the 'Name' property:

```
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Find("IngredientMixer");
mixerObject.Name = "New_PlantViewNode_Name";
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Working with plant view nodes of CPM object instance (RT Unified)

Introduction

You can perform the following tasks related to access plant view nodes of CPM object instances while using TIA Portal Openness:

- Create plant view nodes
- Delete plant view nodes
- Rename plant view nodes

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code to create a plant view node inside a CPM object instance

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Create("ObjectLevel1",
"Plant_Object_Type_1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Create("Mixing");
```

Note

You can create any number of plant view node inside a CPM object

Modify the following program code to create a CPM object instance inside a plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject = mixingNode.PlantViewNodes.Create("ObjectLevel1",
"Plant_Object_Type_1");
```

Note

You can create any number of CPM object instances inside a plant view node

Modify the following program code to remove or delete a plant view node from a plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Find("ObjectLevel1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Find("Mixing");
mixingNode.Delete();
```

Modify the following program code to rename a plant view node of CPM object instance by updating the 'Name' property:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Find("ObjectLevel1",
"Plant_Object_Type_1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Find("Mixing");
mixingNode.Name = "New_PlantViewNode_Name";
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Enumerating interface tags of CPM object instance (RT Unified)

Introduction

You can use the TIA Portal Openness to enumerate a member of a CPM object instance.

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code to enumerate all members of the CPM object instance:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantViewNodes = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodes.Find("MixerObject_1");
PlantObject plantObject = plantViewNode.PlantObject;
PlantObjectInterfaceComposition interfaces = plantObject?. PlantObjectInterfaces;
```

Note

In the above example, the ?.operator refers to null check

Modify the following program code to find an interface tag from interfaces list based on 'Name':

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantNodes = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantNodes.Find("MixerObject_1");
PlantObject plantObject = plantViewNode.PlantObject;
PlantObjectInterface interface = plantObject. PlantObjectInterfaces.Find("Interface_1");
```

Modify the following program code to find an interface tag from interface list based on index:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantViewNodes = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodes.Find("MixerObject_1");
PlantObject plantObject = plantViewNode.PlantObject;
PlantObjectInterfaceComposition interfaces = plantObject.PlantObjectInterfaces;
PlantObjectInterface interface = interfaces[0];
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.2.7 Accessing properties for interface/logging tags of plant object instances (RT Unified)

Accessing/Updating properties of interface tags of CPM plant object instances (RT Unified)

Introduction

You can use the TIA Portal Openness to access and update the properties on interface tags of CPM plant object instance.

The following properties can be accessed on interface tags of CPM plant object instance:

Property Name	Data type	Description	Accessibility
Name	System.String	Name of interface tags	R/W
PlcTag	System.String	PLC tag associated with interface tags	R/W
Connection	System.String	Connection of interface tags	R/W
PlcName	System.String	PLC name associated with interface tags	R
AccessMode	PlantObjectTagAccessMode	Access mode for interface tags	R
DataType	System.String	Object type of the interface tags	R
MaxLength	System.Int	Length of interface tags	R
HmiDataType	System.Int	HMI data type of interface tags	R
AcquisitionMode	PlantObjectTagAcquisitionMode	Acquisition mode of interface tags	R/W
AcquisitionCycle	System.Int	Acquisition cycle for interface tags	R/W
Persistent	System.Bool	Persistence for internal tags of CPM object instance	R
Comment	System.String	Comment	R/W
Members	PlantObjectInterfaceMemberComposition	Members of Interface Tag	R

Note

PLCTag & Connection property can only be configured after assigning Hmi device to the plant view.

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code to access all the properties of interface tags:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
string name = interfaceTag.Name;
MultilingualText comment = interfaceTag.Comment
```

Modify the following program code to update some properties of an interface tag, such as Comment, PLC Tag, Connection, Acquisition Mode, Acquisition Cycle:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
interfaceTag.Name = "New_InterfaceTag_Name";
interfaceTag.Comment.Items[0].Text = "New_InterfaceTag_Comment";
interfaceTag.AcquisitionMode = PlantObjectTagAcquisitionMode.CyclicContinuous;
interfaceTag.AcquisitionCycle = "T12s";
interfaceTag.Connection = "Connection_1";
interfaceTag.PlcTag = "PLC_Tag_1";
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing/Updating properties of members tags of interface tags (RT Unified)

Introduction

You can use the TIA Portal Openness to access and update properties of member tags of interface tags.

The following properties can be accessed on member tags of interface tags:

Property Name	Data type	Description	Accessibility
Name	System.String	Name of member tags	R
DataType	System.String	Object type of the member tags	R
MaxLength	System.Int	Length of member tags	R
HmiDataType	System.String	HMI data type of member tags	R
InitialMinValue	PlantObjectTagLowerRange	Lower range value	R/W
InitialMaxValue	PlantObjectTagUpperRange	Upper range value	R/W
InitialValue	System.Object	Initial value	R/W
SubstituteValue	PlantObjectTagSubstituteValue	Substitute value	R
Comment	MultilingualText	Comment	R/W
Members	PlantObjectInterfaceMemberComposition	Members of Member Tag	R
LoggingTags	PlantObjectLoggingTagComposition	LoggingTags of MemberTag	R

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantViewNodeComposition = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodeComposition[0];
PlantObjectInterfaceComposition tagComposition =
plantViewNode.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = tagComposition[0];
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
//GetAttributes
List<string> listOfPropertyNames = new List<string>()
{
"Name"
};
var listOfPropertyValues = memberTag.GetAttributes(listOfPropertyNames);
//SetAttributes
IEnumerable<KeyValuePair<string, object>> propertyNameValuePairList = new
List<KeyValuePair<string, object>>()
{
new KeyValuePair<string, object>("Name", "MemberTag_1");
};
memberTag.SetAttributes(propertyNameValuePairList);
```

Modify the following program code to access all the properties of member tags:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
string name = memberTag.Name;
string comment = memberTag.Comment.Items[0].Text;
```

Modify the following program code to update properties of a member tag of a CPM object instance:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
memberTag.Comment.Items[0].Text = "New_MemberTag_Comment";
memberTag.InitialValue = 2;
memberTag.InitialMinValue.Value = 2;
memberTag.InitialMinValue.ValueType = PlantObjectTagLimitValueType.Constant;
memberTag.InitialMaxValue.Value = 2;
memberTag.InitialMaxValue.ValueType = PlantObjectTagLimitValueType.Constant;
```

Modify the following code to access all the logging tags of a member tag of CPM object instance:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
PlantObjectLoggingTagComposition loggingTags = memberTag.LoggingTags;
PlantObjectLoggingTag loggingTag = loggingTags.FirstOrDefault();
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Accessing/Updating properties of logging tags of member tags (RT Unified)

Introduction

You can use the TIA Portal Openness to access logging tags of member tags present inside CPM object instance.

The following properties can be accessed on logging tags of member tags present inside CPM object instance:

Property name	Data type	Description	Accessibilty
AggregationDelay	System.TimeSpan	Aggregation delay value	R
AggregationMode	PlantObjectLoggingTagAggregationMode	Aggregation Mode value	R
Cycle	System.String	Logging Cycle of logging tag	R
CycleFactor	System.UInt32	Logging Cycle Factor vlaue	R
DataLog	System.String		R
Source	System.string	Source Logging Tag	R
TriggerMode	PlantObjectLoggingTagTriggerMode	TriggerMode of Logging tag	R
TriggerTag	System.String	TriggerTag Value	R
TriggerTagBitNumber	System.UInt32	TriggerTagBitNumber value	R
Name	System.string	Name of logging tag	R
LogConfiguration	System.String	Data log of logging tag	R
LoggingMode	PlantObjectLoggingTagLoggingMode	Logging mode of logging tag	R
SmoothingMode	PlantObjectLoggingTagSmoothingMode	Smoothing mode of logging tag	R
SmoothingMinTime	System.TimeSpan	Minimum time of logging tag	R
SmoothingMaxTime	System.TimeSpan	Maximum time of logging tag	R
SmoothingDeltaValue	System.Double	Delta of logging tag	R
LimitScope	PlantObjectLoggingTagLimitScope	Limit scope of logging tag	R
HighLimit	System.Object	High limit of logging tag	R
LowLimit	System. Object	Low limit of logging tag	R

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Modify the following program code to access all the properties of logging tags:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceMemberComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaceMembers;
PlantObjectInterfaceMember interfaceTag = interfaceTags.FirstOrDefault();
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
PlantObjectLoggingTagComposition loggingTags = memberTag.LoggingTags;
PlantObjectLoggingTag loggingTag = loggingTags.FirstOrDefault();
string name = loggingTag.Name;
object lowLimit = loggingTag.LowLimit;
```

Note

You will not be able to update any property of PlantObjectLoggingTag class, associated to a CPM object instance logging tag.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.13.2.8 Checking license for access Unified device (RT Unified)

Introduction

You can use the TIA Openness interface to check any Openness client application for the presence of WinCC Unified license, so that a client application cannot bypass licensing using Openness interfaces.

You are only able to browse to an HMI Unified device and all its underlying objects (e.g. tags, alarms, screen, etc.) when HMI Unified license is not available. Any other operation on device level (e.g. create, find, delete) results in a LicenseNotFoundException being returned from Openness API.

Note

For HMI Unified Software, If license is invalid, then a LicenseNotFound Exception will be returned.

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- Access to HMI Software object
  See HMI Unified Software object (Page 653)

Program code

Modify the following program code to access the device or device sub systems when license is not available, a LicenseNotFound Exception is returned:

```
private static void ListDevices(Project project)
{
foreach (var device in project.Devices)
{
ListDeviceItem(device);
}
}

private static void ListDeviceItem(Device device)
{
try
{
Console.Write("HMI device - " + device.Name);
foreach (var item in device.DeviceItems)
{
Console.WriteLine("HMI device type - " + item.TypeIdentifier);
var softContainer = item.GetService<SoftwareContainer>();
if (softContainer != null)
{
var softTarget = softContainer.Software;
}
if (softTarget != null)
{
Console.WriteLine("HMI device software - " + softTarget.Name);
}
}
}
catch (Exception ex)
{
Console.WriteLine(ex.Message);
}
}
```

Note

While accessing the above program code, the softTarget will be null, and accessing sub-systems, such as alarms, tags will not be posssible.

Modify the following program code to access the device or device sub systems when license is available, a null value is returned.

```
private static void ListDevices(Project project)
{
foreach (var device in project.Devices)
{
ListDeviceItem(device);
}
}
private static void ListDeviceItem(Device device)
{
Console.Write("HMI device - " + device.Name);
foreach (var item in device.DeviceItems)
{
Console.WriteLine("HMI device type - " + item.TypeIdentifier);
var softContainer = item.GetService<SoftwareContainer>();
if (softContainer != null)
{
var softTarget = softContainer.Software;
}
if (softTarget != null)
{
Console.WriteLine("HMI device software - " + softTarget.Name);
}
}
}
}
```

Note

While accessing the above program code, the softTarget will have device container, and accessing sub-systems, such as alarms, tags will be possible.

See also

Connecting to the TIA Portal (Page 82)

Description HMISoftware (Page 653)

5.13.2.9 Getting Cross References for WinCC Unified (RT Unified)

Introduction

You can use the TIA Portal Openness to provide cross reference information on applicable WinCC Unified objects. The Openness API returns CrossReferenceResult which provide the textual values of cross reference user interface and actual IEngineeringObject of applicable WinCC Unified objects.

The cross reference will attempt to provide the WinCC Unified object itself on which you can perform additional actions supported by the respective TIA Portal Openness objects.

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- Acess to the HMI Software object
  See HMI Unified Software (Page 653)

Properties

For information on list of properties supported by WinCC Unified objects, Refer Properties in Getting CrossReferences for Step7 (Page 447)

Program code

```
Public void GetCrossReferences()
{
CrossReferenceService crossReferenceService =
HmiScreen.GetService<CrossReferenceService>();
CrossReferenceResult rootResultObject =
crossReferenceService.GetCrossReferences(CrossReferenceFilter.AllObjects);
SourceObjectComposition rootResultSourceObjects = rootResultObject.Sources; //The API
returns a single source object but the type is a composition.
foreach (SourceObject sourceObject in rootResultSourceObjects)
{
//Get the source object references
ReferenceObjectComposition referenceObjects = source.References;
PrintReferences(referenceObjects);
}
}
Private void PrintReferences(ReferenceObjectComposition referenceObjects)
{
foreach (ReferenceObject referenceObject in referenceObjects)
{
// Needs similar casting for each type as seen in source object example snippet
if (source.UnderlyingObject != null && source.UnderlyingObject is HmiScreenItemBase)
{
var hmiScreenItem= source.UnderlyingObject as HmiScreenItem;
//Perform actions on hmiScreenItem.
}
//No other attributes are available from the IEngineeringObject. All the remaining values
must be given from crossreference separately via ReferenceObject EOM.
Console.WriteLine(referenceObject.Name);
Console.WriteLine(referenceObject.Address);
Console.WriteLine(referenceObject.TypeName);
Console.WriteLine(referenceObject.Device);
Console.WriteLine(referenceObject.Path);
LocationComposition locations = referenceObject.Locations;
//Looping through ReferenceLocations
{
Console.WriteLine(location.Name);
Console.WriteLine(location.ReferenceLocation);
Console.WriteLine(location.ReferenceType);
Console.WriteLine(location.Access);
if(location.ReferencedAs is Tag)
{
var referencedAsObject = location.ReferencedAs as Tag;
}
// Needs similar casting for each type as seen in source object example snippet
Console.WriteLine(location.ReferncedAsName);
Console.WriteLine(location.Address);
Console.WriteLine(location.TypeName);
}
}
```

See also

Getting Cross References for Step7 (Page 447)

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Description HMISoftware (Page 653)

5.13.2.10 Working with dynamization & events for screen/screen items using scripts (RT Unified)

Introduction

You can perform the following tasks with script to configure property dynamization and events for screen/screen items while using TIA Portal Openness:

- Creating script dynamizations
- Enumerating script dynamizations
- Deleting script dynamizations
- Accessing properties of script dynamizations
- Accessing properties of trigger
- Checking syntax

Requirement

- The TIA Portal Openness application is connecte to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- Access to HMI Uinified Software object
  See HMI Unified Software object (Page 653)

Program code

You can modify and use the following program code examples while working with script dynamization for screen/screen items.

Creating script dynamizations

Modify the following program code to create script dynamizations for a property:

```
public DynamizationBase Create<DynamizationBase>(string propertyName);
HmiScreen screen = m_hmisoftware.Screen[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
ScriptDynamization scriptDynamic = dyns.Create<ScriptDynamization>("BackColor");
```

Enumerating script dynamizations

Modify the following program code to enumerate script dynamization of a property:

```
public DynamizationBase Find(string propertyName);
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
if (dyns is ScriptDynamization)
{
ScriptDynamization scriptDynamic = (ScriptDynamization)dyns.Find("BackColor");
}
```

Deleting script dynamizations

Modify the following program code to delete script dynamization for a property:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
if (dyns is ScriptDynamization)
ScriptDynamization scriptDynamic = (ScriptDynamization)dyns.Find("BackColor");
scriptDynamic.Delete();
```

Accessing properties of script dynamizations

Modify the following program code to get and set properties of script dynamization:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
foreach (DynamizationBase dynamic in dyns)
{
if (dynamic.DynamizationType == DynamizationType.Script)
{
ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
scriptDynamization.Async = true;
scriptDynamization.GlobalDefinitionAreaScriptCode = @"Add(parameter1,parameter2)";
scriptDynamization.ScriptCode = @"
var value;
let tag1 = Tags('Tag_1');
let tagValue1 = tag1.Read();
HMIRuntime.Trace('value of MyTag1: ' + tagValue1);
return value; ";
Trigger triggerObj = scriptDynamization.Trigger;
}
}
```

Accessing properties of Trigger

Modify the following program code to get and set properties of trigger:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
foreach (DynamizationBase dynamic in dyns)
{
if (dynamic.DynamizationType == DynamizationType.Script)
{
ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
// Property write
scriptDynamization.Trigger.Type = TriggerType.CustomCycle;
scriptDynamization.Trigger.CustomDuration = "T500ms";
scriptDynamization.Trigger.Tags = new List<string>() { "Tag_1" };
// Property read
Console.WriteLine(scriptDynamization.Trigger.Type);
Console.WriteLine(scriptDynamization.Trigger.CustomDuration);
Console.WriteLine(scriptDynamization.Trigger.Tags);
}
}
```

Syntax Check

Modify the following program code to check syntax for script code or global script code using SyntaxCheck action (method) of script dynamization:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
foreach (DynamizationBase dynamic in dyns)
{
if (dynamic.DynamizationType == DynamizationType.Script)
{
ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
scriptDynamization.ScriptCode = @"
var value;let tag1 = Tags('Tag_1');
let tagValue1 = tag1.Read();
HMIRuntime.Trace('value of MyTag1: ' + tagValue1);
return value; ";
HmiValidationResult syntaxCkResult = scriptDynamization.SyntaxCheck();
IEnummerable<string> error = syntaxCkResult.Errors;
IEnumerable<string> warnings = syntaxCkResult.Warning;
}
}
```

Note

Create - Recoverable exception will be thrown in case respective dynamization is not supported for the specified property. If during set of properties the set operation fails consistency checks then a Recoverable exception would be raised.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Description HMISoftware (Page 653)

5.13.2.11 Working with dynamization for screens/screen items (RT Unified)

Introduction

You can perform the following tasks with dynamization feature of screens/screen items while using TIA Portal Openness:

- Creating dynamization for a property
- Enumerating dynamization for a property
- Deleting dynamization for a property
- Accessing common properties of dynamization
- Accessing properties of tag dynamization
- Accessing properties of flasing dynamization
- Accessing properties of resourcelist dynamization

Note

For accessing dynamization for a properties at part level, you need to access respective part of Control. For part information, see Accessing properties of trend control (Page 82)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A Project is open
  See Opening a project (Page 128)
- Access to HMI Software object
  See HMI Unified Software object (Page 653)

Program code

You can modify and use the following program code examples while working with dynamizations for screen/screen items.

Creating dynamization for a property

Modify the following program code to create dynamization for a property:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
// Flashing dynamization
FlashingDynamization FlashingDyn = dyns.Create<FlashingDynamization>("BackColor");
// ResourceList dynamization
ResourceListDynamization ResListDyn = dyns.Create<ResourceListDynamization>("DisplayName");
// Tag dynamization
TagDynamization tagDyn = dyns.Create<TagDynamization>("Width");
```

Enumerating dynamization of a property

Modify the following program code to enumerate dynamization for a property:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
// Flashing dynamization
FlashingDynamization FlashDyn = (FlashingDynamization)dyns.Find("BackColor");
// ResourceList dynamization
ResourceListDynamization resourceDyn = (ResourceListDynamization)dyns.Find("DisplayName");
// Tag dynamization
TagDynamization tagDyn = (TagDynamization)dyns.Find("Width");
```

Deleting dynamization of a property

Modify the following program code to delete dynamization of a property:

```
public void Delete ()
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
DynamizationBase dynamization = dyns.Find("BackColor");
dynamization.Delete();
```

Accessing common properties of dynamization

Modify the following program code to get common properties of dynamization:

```
HmiSoftware m_hmiSoftware = ..;
DynamizationBaseComposition screenDynamizations = m_hmiSoftware.Screens[0].Dynamizations;
DynamizationBase screenDynamization = screenDynamizations.Find("BackColor");
string propertyName = screenDynamization.PropertyName;
DynamizationType dynamizationType = screenDynamization.DynamizationType;
```

Accessing properties of tag dynamization

Modify the following program code to get and set properties of tag dynamization:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
TagDynamization tagDynamization = (TagDynamization)screen.Dynamizations.Find("Width");
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.Tag)
{
TagDynamization TagDynamization = (TagDynamization)dynamization;
TagDynamization.Tag = "HmiTagName";
TagDynamization.UseIndirectAddressing = true;
TagDynamization.ReadOnly = true;
}
}
```

Accessing properties of flashing dynamization

Modify the following program to get and set properties of flashing dynamization:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.Flashing)
{
FlashingDynamization flashingDynamization = (FlashingDynamization)dynamization;
flashingDynamization.AlternateColor = Color.Beige;
flashingDynamization.Color = Color.FromArgb(12, 22, 152, 200);
flashingDynamization.FlashingCondition = FlashingCondition.Always;
flashingDynamization.FlashingRate = FlashingRate.Medium;
}
}
```

Accessing properties of resourcelist dynamization

Modify the following program code to get and set properties of resourcelist dynamization

```
HmiScreen screen = m_hmiSoftware.Screens[0];
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.ResourceList)
{
ResourceListDynamization resourceListDyn = (ResourceListDynamization)dynamization;
resourceListDyn.Tag = "Tag name";
resourceListDyn.ResourceList = "Resource list name";
}
}
```

Note

Create - Recoverable exception will be thrown in case respective dynamization is not supported for the specified property. If during set of properties the set operation fails consistency checks then a Recoverable exception would be raised.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Description HMISoftware (Page 653)

5.14 Functions for Version Control Interface

5.14.1 Accessing VCI system group in project

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access the VCI system group via VersionControlInterface from the project as a service. The VersionControlInterface is a service and returns non-null object.

Program code

Modify the following program code to retrieve the workspace system group from the VersionControlInterface:

```
public void function01()
{
//Accessing VCI system group in project
Siemens.Engineering.Project project = tiaPortal.Projects[0];
Siemens.Engineering.VersionControl.VersionControlInterface versionControlInterface =
project.GetService<VersionControlInterface>();
Siemens.Engineering.VersionControl.WorkspaceSystemGroup workspaceSystemGroup =
versionControlInterface.WorkspaceGroup;
}
```

5.14.2 Enumerating user groups in VCI group

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to enumerate all VCI workspace user groups in other VCI workspace groups.

```
WorkspaceUserGroup workspaceGroup = ...;
WorkspaceUserGroupComposition userGroupComposition = workspaceGroup.Groups;
foreach (Siemens.Engineering.VersionControl.WorkspaceUserGroup workspaceUserGroup in
userGroupComposition)
{
//...
}
```

Modify the following program code to access individual workspace user group in other VCI workspace groups:

```
WorkspaceUserGroup workspaceGroup = ...;
Siemens.Engineering.VersionControl.WorkspaceUserGroup workspaceUserGroup =
workspaceGroup.Groups.Find("Some Group Name");
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.14.3 Creating VCI user group in VCI group

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to create a workspace user group in a workspace group.

Program code

Modify the following program code to create a workspace user group:

```
Project project = ...;
//Creating VCI user group in VCI group
VersionControlInterface versionControlInterface =
project.GetService<VersionControlInterface>();
WorkspaceSystemGroup workspaceGroupComposition = versionControlInterface.WorkspaceGroup;
WorkspaceUserGroup result =
workspaceGroupComposition.Groups.Create("NewWorkspaceUserGroup");
```

Note

To create a new workspace user group by name requires that supplied name should be valid and the supplied name does not already exist on another workspace user group

5.14.4 Updating VCI group properties

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to update the properties of workspace user group.

The following property is available to update on a workspace group:

Property Name	Return Type	Description	Accessibility
Name	System.String	The name of the workspace user group	Read/Write

Program code

Modify the following program code to update workspace user group name:

```
//Updating VCI group properties
var workspaceUserGroup = ...;
workspaceUserGroup.Name = "New_Group_Name";
```

5.14.5 Deleting VCI user group

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to delete workspace user groups. When you are deleting workspace user groups, all other objects contained in the workspace user group will also get deleted. This includes other workspace user groups and VCI workspaces.

Program code

Modify the following program code to delete a workspace user group:

```
//Deleting VCI user group
WorkspaceUserGroup workspaceUserGroup = ...;
workspaceUserGroup.Delete();
```

5.14.6 Enumerating VCI workspaces in VCI group

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a Project (Page 128)

Program code

Modify the followng program code to enumerate all VCI workspaces in a VCI group:

```
WorkspaceSystemGroup workspaceSystemGroup = ...;
WorkspaceComposition workspaceComposition = workspaceSystemGroup.Workspaces;
foreach (Siemens.Engineering.VersionControl.Workspace workspace in workspaceComposition)
{
//...
}
```

Modify the following program code to find specific workspace by name:

```
WorkspaceUserGroup workspaceUserGroup = ...;
//Enumerating VCI workspaces in VCI group 02
Siemens.Engineering.VersionControl.Workspace workspace =
workspaceUserGroup.Workspaces.Find("SomeWorkspaceName");
```

See also

Connecting to the TIA Portal (Page 82)

5.14.7 Creating VCI workspace in VCI group

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to create a workspace in a workspace user group.

You can create Siemens.Engineering.VersionControl.Workspace with the create action:

```
Siemens.Engineering.VersionControl.WorkspaceComposition.Create(strin
g name)
```

Program code

Modify the following program code to create a workspace:

```
Project project = ...;
WorkspaceUserGroup workspaceUserGroup = ...;
//Creating VCI workspace in VCI group
VersionControlInterface versionControlInterface =
project.GetService<VersionControlInterface>();
WorkspaceComposition workspaceComposition =
versionControlInterface.WorkspaceGroup.Workspaces;
Workspace result = workspaceComposition.Create("NewWorkspace");
```

Note

To create a new workspace by name requires that supplied name should be valid and the name does not already exist on another workspace user group.

5.14.8 Updating VCI workspace properties

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to update the properties of a VCI workspace.

The following properties are available to update on a workspace:

Property name	Return type	Description	Accessibility
Name	System.String	The name of the workspace.	Read/Write
RootPath	System.IO.DirectoryInfo	The configured workspace path.	Read/Write

Program code: Updating name property

Modify the following program code to update the 'Name' property on workspace:

```
var workspace = ...;
workspace.Name = "New_Workspace_Name";
```

Program code: Updating rootpath property

Modify the following program code to configure the workspace path to a rooted directory info. Setting this value to null will unconfigure the workspaces root path.

```
var workspace = ...;
workspace.RootPath = new DirectoryInfo(@"D:\Project_WS");
```

Modify the following program code to unconfigure a workspace:

```
var workspace = ...;
workspace.RootPath = ...;
```

5.14.9 Deleting VCI workspace

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to delete a workspace. When a workspace is deleted all of the workspace's mappings get also deleted.

Program code

Modify the following program code to delete workspace:

```
//Deleting VCI workspace
Workspace workspace = ...;
workspace.Delete();
```

5.14.10 Enumerating VCI workspace mappings in VCI

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to enumerate all VCI workspace mappings in a workspace:

```
var workspace = ...;
WorkspaceMappingComposition mappings = workspace.Mappings;
foreach (Siemens.Engineering.VersionControl.WorkspaceMapping workspaceMapping in mappings)
{
//...
}
```

Modify the following program code to access individual workspace mapping for the specific engineering object:

```
PlcBlock block = ...;
var workspace = ...;
IEngineeringObject versionControlSupportedEngineeringObject = block;
Siemens.Engineering.VersionControl.WorkspaceMapping workspaceMapping =
workspace.Mappings.Find(versionControlSupportedEngineeringObject);
```

Note

The IEngineeringObject interface is derived from IEngineeringCompositionObject and IEngineeringInstance. For information on IEngineeringObject, refer Dynamic behaviours (Page 106)

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.14.11 Creating VCI workspace mapping in VCI workspace

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to create a workspace mapping in VCI workspace.

You can create Siemens.Engineering.VersionControl.WorkspaceMapping with the create action signature:

```
Siemens.Engineering.VersionControl.WorkspaceMappingComposition.Creat
e(string relativeWorkspacePath, IEngineeringObject
linkedProjectObject)
```

Program code

Modify the following code to create a workspace mapping:

```
//Creating VCI workspace mapping in VCI workspace
var workspace = ...;
var plcBlock = ...;
var result = workspace.Mappings.Create(@"\TestCopy\Block_1.xml", plcBlock);
```

The Recoverable Exception will be thrown in case of :

- The linked object is not a valid VCI supported object
- The linked object is already mapped.

- The relative file path contains invalid file characters.
- The relative file path contains parent directory navigation.

5.14.12 Updating VCI workspace mapping properties

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Openness Portal to update the properties of a VCI workspace mapping.

The following properties are available on a workspace mapping.

Property name	Return type	Description	Accessibility
RelativeWorkspacePath	System.String	The relative file path to the linked objects representation in source control.	Read/write
LinkedProjectObject	IEngineeringObject	The engineering object compatible with VCI that is linked to the file in source control	Read

Program code

Modify the following program code to update the workspace mapping relative file path:

```
//Updating VCI workspace mapping properties
var workspaceMapping = ...;
workspaceMapping.RelativeWorkspacePath = @"\Test\NewBlock_1.xml";
```

5.14.13 Deleting VCI workspace mapping

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to delete a VCI workspace mapping.

Program code

Modify the following program code to delete a VCI workspace mapping:

```
//Deleting VCI workspace mapping
WorkspaceMapping workspaceMapping = ...;
workspaceMapping.Delete();
```

5.14.14 Synchronizing a workspace mapping

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to view the status of synchronization workspace mapping.

Program code: Individual Object Synchronization status

You can view status, update status and synchronize status of individual objects you must first request for Siemens.Engineering.VersionControl.WorkspaceMapping.IndividualObjectSynchronizationStatus from workspace mapping as work

It is recommended to perform a null check before invoking the actions on individual object synchronization service as that mapping may or may not support individual object synchronization.

```
var individualObjectSynchronizationStatus =
workpaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null);
{
 //GetStatus()...
 //UpdateStatus()..
 //Synchronize()..
}
```

Program code: Get Status

You can view the synchronization status of inidividual object by using the action Siemens.Engineering.VersionControl.IndividualObjectSynchronizationStatus.GetStatus() which returns Siemens.Engineering.VersionControl.IndividualObjectCompareResult.

The IndividualObjectCompareResult object is used to inform about the current sync status.

```
public class IndividualObjectCompareResult
{
CompareState CompareState { get; }
IndividualObjectCompareDetails {get; }
}
```

The IndividualObjectCompareDetails flag enum is used to inform you about which part of the workspace mapping has changed (or both). This value will be None when the

CompareState is Equal. If CompareState is unequal, this enum can have a value of ProjectObjectChanged or WorkspaceFileChanged or both ProjectObjectChanged**,**

WorkspaceFileChanged.

```
[Flags]public enum IndividualObjectCompareDetails
{
None = 0,
ProjectObjectChanged = 1,
WorkspaceFileChanged = 2
}
```

The CompareState enum is used to inform if an object has any changes.

```
public enum CompareState
{
Equal,
Unequal,
WorkspaceFileMissing,
Unknown
}
```

Modify the following program code to get the status of a workspace mapping:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workpaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null)
{
var compareResult = individualObjectSynchronizationStatus.GetStatus();
//when compareState is unequal
{
if(compareResult.CompareState == CompareState.Unequal)
{
if(compareResult.IndividualObjectCompareDetails ==
IndividualObjectCompareDetails.WorkspaceFileChanged)
{
//Workspace file has changed
}
elseif(compareResult.IndividualObjectCompareDetails ==
(IndividualObjectCompareDetails.ProjectObjectChanged |
IndividualObjectCompareDetails.WorkspaceFileChanged))
{
//Both project object and workspace file has changed
}
}
}
```

Program code: Updating synchronization status

You can force the system to perform a live comparison between the currently linked project object and workspace file. This can result in a modification to sync status to reflect the current state of the system, and can be used to determine the status of a recently linked (but not synchronized) mapping without modifying either the existing project object or workspace file.

Modify the following program code to update the status of a workspace mapping:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workpaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != Null)
{
individualObjectSynchronizationStatus.UpdateStatus();
}
```

Program code: Synchronizing workspace

You can force the system to perform a live comparison between the currently linked project object and workspace file. This can result in a modification to sync status to reflect the current state of the system, and can be used to determine the status of a recently linked (but not synchronized) mapping without modifying either the existing project object or workspace file.

The SynchronizationMode enum is used to inform the system which direction to perform the sync:

```
public enum SynchronizationMode
{
ProjectToWorkspace,
WorkspaceToProject
}
```

This action will attempt to sync the linked project object to/from the linked workspace file object.

Modify the following program code to synchronize a workspace:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workpaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null)
{
individualObjectSynchronizationStatus.Synchronize(SynchronizationMode.ProjectToWorkspace);
}
```

5.14.15 Accessing synchronization status of child objects

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to get the synchronization status of the child objects.

Method

Method Name	Description
GetService<ChildObjectsStateProvider>()	Returns the ChildObjectsStateProvider service object from workspace mapping on the object which has one or more VCI enabled child objects.

The ChildObjectStateProvider object provides the following method to view the synchronization status of child objects.

Method Name	Description
GetStatus()	Returns the SynchronizationResult object representing the current synchronization status of child objects.

ENUM

The SynchronizationResult object provides the following Enum to view if an object has any changes.

Enum	Value	Description
MappingState	Equal	All the child objects are mapped and in synchronization.
	UnEqual	A child object of a hierarchial object is out of synchronization.

Program code

```
var workspaceMapping = ...;
//To get the ChildObjectStateProvider object from the worksspace mapping that contains the
child of hierarchial objects.
var childObjectsStateProvider = workpaceMapping.GetService<ChildObjectsStateProvider>();
//To perform null check before invoking the actions on child object synchronization service
as the mapping may or may not support child objects synchronization.
if(childObjectsStateProvider != null)
{
//The GetStatus()returns all the child objects are mapped and are in sync.
//The SynchronizationResult returns the mapping status as Equal.
var synchronizationResult= childObjectsStateProvider.GetStatus();
//When MappingState is unequal
if(synchronizationResult.MappingState == MappingState.Unequal)
{
//Access to an individual child object and retrieve the status
}
}
```

5.14.16 Creating VCI workspace with workspace language

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the Workspace.Create() in TIA Portal Openness to create a workspace in version control interface (VCI) using the following ways:

- Create a workspace with the workspace language set to the default TIA Portal user interface language and setting the root path as a parameter in the Workspace.Create() API.
- Create a workspace with the workspace language by assigning the workspace language and the root path as a paramter in the Workspace.Create() API.

Note

In the TIA Portal Openness V18 and previous versions you can create a workspace with the workspace name. The language will be implicitly set to the default TIA Portal user interface language and the root path must be explicitly set after the Workspace.Create() API call.

Parameter

The Workspace.Create() accepts the following parameters to create a workspace with workspace language in VCI:

Parameter name	Datatype	Description
Name	String	Specifies the name of the workspace to be created
RootPath	DirectoryInfo	Specifies the location of the path to configure the workspace
WorkspaceLanguage	System.Globalization.CultureInfo	Specifies the configured workspace language

Method

The Workspace.Create() creates the workspace in VCI:

Method name	Description
Workspace.Create(string name, DirectoryInfo rootPath, CultureInfo workspaceLanguage)	Creates a new workspace with specified name, root path and workspace language culture
Workspace.Create(string name, DirectoryInfo rootPath)	Creates a new workspace with specified name and the root path. The default TIA Portal user interface language is assigned to the workspace language.
Workspace.Create(string name)	Creates a new workspace with specified name. The default TIA Portal user interface language is assigned to the workspace language and the root path is to be set explicitly by the user.

Program code

To create a workspace with specified workspace name and language set to the default TIA Portal user interface language. The root path is to be set explicitly by the user modify the following program code:

```
using Siemens.Engineering.VersionControlInterface;
private void createWorkspace()
{
Workspace workspace = WorkspaceGroup.Workspaces.Create("worskspace1");
workspace.RootPath = new DirectoryInfo(Directory.GetCurrentDirectory());
}
```

To create a workspace with workspace language set to the default TIA Portal user interface language and the root path to be set as a parameter in the workspace.create() modify the following program code:

```
using Siemens.Engineering.VersionControl;
private void createWorkspace()
{
WorkspaceComposition workspace = WorkspaceGroup.Workspaces.Create("workspace1"), new
DirectoryInfo(Directory.GetCurrentDirectory()));
}
```

To create a workspace with workspace language via assigning the workspace language and the root path as a parameter in the workspace.create() modify the following program code:

```
using Siemens.Engineering.VersionControl;
private void createWorkspaceLanguage()
{
Workspace workspace = WorkspaceGroup.Workspaces.Create("workspace1", new
DirectoryInfo(Directory.GetCurrentDirectory()), new CultureInfo("de-DE"));
}
```

Note

An "EngineeringTargetInvocationException" exception is thrown in the following cases:

- Invalid workspace language
- Set a workspace language in the workspace.create() API when the workspace language is already available in the root path.

5.14.17 Initializing VCI object status

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to initialize the status of the VCI object to the mapped file. The API can be used only if you are certain that the files and project objects are not modified.

The API is available through the IndividualObjectSynchronizationStatus service as a part of the workspace mapping and returns green as a successful result for VCI status.

The EngineeringTargetInvocationException exception will be thrown when the VCI status is unknown for example n ("?" in UI)".

Method

Method name	Description
public void InitializeStatus()	Initializes the status of the VCI object and the mapped workspace file and sets the status to Green/Equal.

Program code

```
var workspaceMapping=...;
var individualObjectSynchronizationStatus =
workpaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if (individualObjectSynchronizationStatus != null)
{
   var currentStatus = individualObjectSynchronizationStatus.GetStatus().CompareState;
   if (currentStatus.Equals(CompareState.Unknown))
   {
     individualObjectSynchronizationStatus.InitializeStatus();
   }
}
```

5.14.18 Importing Simatic ML with inactive culture

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the VCI Import setting options to automatically activate the inactive project languages (inactive cultures) in the project when a VCI-import is triggered using a Simatic ML file having a language culture which is not activated in the TIA Project.

You can also access TIA Portal user interface setting using Openness via TiaPortalSettingsFolder class. The TiaPortalSettingsFolder class will be accessible via the Settings property on the TiaPortal class.

Property

The VCI settings will be provided via the TiaPortalSettingsFolder class. The TiaPortalSettingsFolder instance will have the name VCI.

Setting name	Setting data-type	Description	Access
SimaticMLImportOptions	System.Int32	Activates Inactive culture/language during SimaticMlimport or blocks import if the culture/ language is not active.	R/W

SimaticMlImportOptions is a number enum defined in the VersionControl namespace with the below values:

Enum options	Value	Description
SimaticMlImportOptions.ActivateInactiveCultures	1	Activate the cultures if cultures are inactive in TIA Portal project
SimaticMlImportOptions.DoNotActivateInactiveCultures	0	Not Activate the cultures if cultures are inactive in TIA Portal project

Program code

```
...

TiaPortalSettingsFolder vciSettingsFolder = TiaPortal.SettingsFolders.Find("VCI");
TiaPortalSetting inactiveCultureSetting =
vciSettingsFolder.Settings.Find("SimaticMlImportOptions");
// For Activating the inactiveCultures:
inactiveCultureSetting.Value = SimaticMlImportOptions.ActivateInactiveCultures;
//
For Not Activating the inactiveCultures:
inactiveCultureSetting.Value = SimaticMlImportOptions.DoNotActivateInactiveCultures;
```

5.15 Functions support for Multiuser

5.15.1 Creating project server connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Introduction

You can use the Create() to create a new project server connection entry in the TIA Portal settings. The Create() returns the ProjectServer and accepts the following parameters:

Parameter	Data type	Description
aliasName	string	Specifies the alias name for creating project server connection
protocol	Protocol	Specifies the protocol type for creating project server connection
hostname	string	Specifies the host name for creating project server connection
port	int	Specifiies the port number for creating project server connection

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
Protocol protocol = Protocol.Https;
string hostName = "localhost";
int port = 16000;
ProjectServer projectServer = tiaPortal.ProjectServers.Create(aliasName, protocol,
hostName, port);
```

See also

Opening a project (Page 128)

5.15.2 Editing project server connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Introduction

You can use the SetProtocol(), SetHostName(), and SetPort() in TIA Portal Openness to edit an existing project server connection in the TIA Portal settings. For example you can use the API to edit protocol, hostname, and port parameters.

Note

If alias name has to be modified, then the project server connection entry has to be deleted and a new connection has to be created.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
projectServer.SetProtocol(Protocol.Http);
projectServer.SetHostName("RemoteMachineHostName");
projectServer.SetPort(17000);
```

See also

Opening a project (Page 128)

5.15.3 Deleting project server connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Introduction

You can use the DeleteConnection() to delete the project server connection entry from TIA Portal settings.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
projectServer.DeleteConnection();
```

See also

Opening a project (Page 128)

5.15.4 Getting project server connection from TIA Portal Setting

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access connections to the project server from TIA Portal Settings. The Openness API will position the connections to the project server as direct navigator of TiaPortal. Connection to the project server are aggregated using the "ProjectServers" composition of the TIA Portal.

Each instance of ProjectServer represents project server connection configuration. Using the instance of ProjectServer retrieved from "ProjectServers" composition, you can comfortably work with the TIA Project server.

Any functionality which involves Multiuser server interactions without opening a local session can be invoked using this navigator and ProjectServers will be available all the time.

The local windows user will be used to authenticate with the multiuser server.

Program code

You can use the below code example to access ProjectServer related properties:

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
// The project server has ServerName property, referred to the "alias" assigned as the name
for the project server is used as an
identifier.
// For example,if full URL of server is "net.tcp://localhost:9876/"
// the projectServer.Host returns "net.tcp://localhost/" and
// projectServer.Port returns 9876
string ServerName = projectServer.ServerName;
string host = projectServer.Host;
int portNumber = projectServer.Port;
```

5.15.5 Adding a project to server

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the AddProjectToServer() to add a single-user project to multiuser server. The AddProjectToServer() accepts the following parameter for the project that is getting added is already opened, then it will be closed.

Parameter	Data type	Description
projectFileInfo	FileInfo	Specifes the path for the server path

Note

For all APIs the local windows user who is running the Openness application/script will be used to authenticate with the multiuser server. It is important that the user is added to the multiuser server with the required role for the intended operation. If the user is not authenticated then an exception will be thrown.

Program code

Modify the following program code to add a single user project to multiuser server:

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
FileInfo projectFileInfo = new FileInfo("C:\\Projects\\Project1\\Project1.ap17");
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
projectServer.AddProjectToServer(projectFileInfo);
```

5.15.6 Getting server projects

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A Project is open
  See Opening a project (Page 128)

Application

You can use the GetServerProjects() to get a list of available server projects info on the project server. The GetServerProjects() returns IEnumerable<ServerProjectInfo>.

You can use this information to perform operations such as:

- Retrieve list of local session info for a specific server project
- Create a local session for a specific server project

Program code

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;


using Siemens.Engineering.SW.Tags;


using Siemens.Engineering.SW.Types;

using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
using Siemens.Engineering.Multiuser;
namespace GettingProjectServer
{
internal class Program
{
private static void Main(string[] args)
{
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects()
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.15.7 Getting available local sessions

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can use the GetLocalSessions() to get a list of available session information for a given server project in the current machine created by the current user.

You can use the information to perform further operations such as opening or deleting the local session.

The GetLocalSessions() returns IEnumerable<ServerProjectInfo> and has serverProjectInfo parameter, whose data type is ServerProjectInfo.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
IEnumerable<LocalSessionInfo> localSessionsInfo =
projectServer.GetLocalSessions(serverProjectInfo[0]);
```

See also

Opening a project (Page 128)

5.15.8 Creating a local session

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is opening
  See Opening a project (Page 128)

Application

You can use the CreateLocalSession() to create a local session for a given server project and to return the LocalSessionInfo object.

The CreateLocalSession() accepts the following parameter for creating a local session using TIA Portal Openness:

Parameter	Data type	Description	Note
serverProjectInfo	ServerProjectInfo	Specifies the list of available projects on server projects	
localSession-Name	String	Speciies the name of local session	
localSessionPath	DirectoryInfo	Specifies the directory path on which the local session is created	
mode	SessionCreation-Mode	The SessionCreationMode accepts the following types of ENUM values: • SessionCreationMode.Multiuser • SessionCreationMode.Exclusive	The server project is automatically locked if the created session is Exclusive local session

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo =
projectServer.CreateLocalSession(serverProjectInfo.First(), localSessionName,
directoryInfo,
SessionCreationMode.Multiuser);
LocalSessionInfo localSessionInfo =
projectServer.CreateLocalSession(serverProjectInfo.First(), localSessionName,
directoryInfo,
SessionCreationMode.Exclusive);
```

5.15.9 Saving a local session

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the Save() to save the given session. The Save() returns the void type and will save the session only if the session is of Engineering or Exclsuive session types. Otherwise, the MultiuserException will be thrown if the session is a server project.

Program code

```
// ...
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo =
projectServer.CreateLocalSession(serverProjectInfo.First(), localSessionName,
directoryInfo,
SessionCreationMode.Exclusive);
LocalSession localSession = tiaPortal.LocalSessions.Open(localSessionInfo.ProjectFileInfo);
//Edit project
localSession.Save();
// ...
```

5.15.10 Deleting local session from server

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the DeleteLocalSessionOnServer() to delete a given local session. The DeleteLocalSessionOnServer() will delete the session from the multiuser server only. The session files in the local disk will not be deleted. The Server project lock is automatically removed if the specified session is an Exclusive session.

The DeleteLocalSessionOnServer() accepts the following parameter to delete local session:

Parameter	Data type	Description
serverProjectInfo	ServerProjectInfo	Specifies the list of available projects on server projects
localsessioninfo	LocalSessionInfo	Specifies the local session information for project

Program code

```
//...
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo =
projectServer.CreateLocalSession(serverProjectInfo.First(), localSessionName,
directoryInfo);
string serverAlias = serverProjectInfo[0].ServerAlias;
string projectName = serverProjectInfo[0].ProjectName;
int sessionId = localSessionInfo.SessionId;
projectServer.DeleteLocalSessionFromServer(serverProjectInfo, localSessionInfo);
//...
```

5.15.11 Opening local/exclusive session

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the Open() to open a multiuser local session or exclusive local session for a given server project. The Open() returns LocalSession object of type engineering session or exclusive session and has a localSessionPath parameter, whose data type is FileInfo.

The server project is automatically locked for the session opened as an Exclusive session .

Program code

Modify the following program code to open Engineering Session:

```
:
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo = projectServer.CreateLocalSession(serverProjectInfo[0],
localSessionName, directoryInfo);
LocalSession multiuserLocalSession =
tiaPortal.LocalSessions.Open(localSessionInfo.ProjectFileInfo);
:
```

Modify the following program code to open Exclusive Session:

```
:
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo = projectServer.CreateLocalSession(serverProjectInfo[0],
localSessionName, directoryInfo, SessionCreationMode.Exclusive);
LocalSession exclusiveSession =
tiaPortal.LocalSessions.Open(localSessionInfo.ProjectFileInfo);
:
```

See also

Opening a project (Page 128)

5.15.12 Opening server project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the OpenServerProject() to open a server project for a given local session. The OpenServerProject() returns the LocalSession object of type server project and has localSessionPath parameter, whose data type is FileInfo.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo =
projectServer.CreateLocalSession(serverProjectInfo.First(), localSessionName,
directoryInfo,
SessionCreationMode.Exclusive);
LocalSession serverProject =
tiaPortal.LocalSessions.OpenServerProject(localSessionInfo.ProjectFileInfo);
```

See also

Opening a project (Page 128)

5.15.13 Checking marked objects

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Introduction

You can use the TIA Portal Openness to provide information if the given EngineeringObject is markable. The TIA Portal Openness also provides marking related information like

- If the object is marked by the current user
- If the object marked by other users
- If the object is up to date

You can also use TIA Portal Openness to get the mark state information of an available EngineeringObject.

The following method and parameter name can be accessed using TIA Portal Openness:

Return type	Method name	Parameters
MarkStateInfo	GetMarkStateInfo	IEngineeringObject engineeringObject

Program code

```
//Open a local session or a server project
//Add a block or edit an block of type IEngineeringObject
MarkStateInfo markStateInfo = localSession.MarkingService.GetMarkStateInfo(muBlock);
if (markStateInfo.IsMarkable)
{
// The object is markable
// Now we can check the markstate for more information
if (markStateInfo.MarkState.HasFlag(MarkState.IsUptoDate))
{
// You have the latest object
}
if (markStateInfo.MarkState.HasFlag(MarkState.IsMarkedByMe))
{
// You have the marked the object
}
if (markStateInfo.MarkState.HasFlag(MarkState.IsMarkedByOthers))
{
// Some other user has marked the object
}
//Example for Checkin without conflicts
if (markStateInfo.MarkState.HasFlag(MarkState.IsUptoDate) &&
markStateInfo.MarkState.HasFlag(MarkState.IsMarkedByMe) && !
markStateInfo.MarkState.HasFlag(MarkState.IsMarkedByOthers))
{
// You have the latest object, the object is marked by me and no one else has marked the
object
}
}
else
{
// The object is not markable. Open server project and make changes, then commit.
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.15.14 Closing local session

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the Close() to close server project, multiuser session, and exclusive session. The Close() returns void type and will discard all pending changes.

Program code

Modify the following program code to close server project:

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo = projectServer.CreateLocalSession(serverProjectInfo[0],
localSessionName, directoryInfo);
LocalSession serverProject =
tiaPortal.LocalSessions.OpenServerProject(localSessionInfo.ProjectFileInfo);
serverProject.Close();
```

Modify the following program code to close exclusive session:

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "serverAliasName";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "ProjectServer1";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo = projectServer.CreateLocalSession(serverProjectInfo[0],
localSessionName, directoryInfo, SessionCreationMode.Exclusive);
LocalSession exclusiveSession =
tiaPortal.LocalSessions.Open(localSessionInfo.ProjectFileInfo);
exclusiveSession.Close();
```

Modify the following program code to close Engineering Session:

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo = projectServer.CreateLocalSession(serverProjectInfo[0],
localSessionName, directoryInfo);
LocalSession multiuserLocalSession =
tiaPortal.LocalSessions.Open(localSessionInfo.ProjectFileInfo);
multiuserLocalSession.Close();
```

5.15.15 Commiting server project changes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the CloseAndCommit() to save the changes done on the server project and exclusive session. The CloseAndCommit() returns the integer type and has Comment parameter of type string.

Program code

Modify the following program to save server project changes:

```
:
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo = projectServer.CreateLocalSession(serverProjectInfo[0],
localSessionName, directoryInfo);
LocalSession serverProject =
tiaPortal.LocalSessions.OpenServerProject(localSessionInfo.ProjectFileInfo);
int revisionCreated = serverProject.CloseAndCommit(comment: "Comment");
:
```

Modify the following program code to save Exclusive Session changes:

```
:
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo = projectServer.CreateLocalSession(serverProjectInfo[0],
localSessionName, directoryInfo, SessionCreationMode.ExclusiveSession);
LocalSession exclusiveSession =
tiaPortal.LocalSessions.Open(localSessionInfo.ProjectFileInfo);
int revisionCreated = exclusiveSession.CloseAndCommit(comment: "Comment");
:
```

5.15.16 Getting lockstate provider

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the GetLockStateProvider() to get information about server project lock and the lock owner. The GetLockStateProvider() returns LockStateProvider type and has serverProjectInfo parameter, whose data type is ServerProjectInfo.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
LockStateProvider lockStateProvider =
projectServer.GetLockStateProvider(serverProjectInfo[0]);
```

5.15.17 Checking project locked

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the IsProjectLocked() to check if the server project is locked or not. The IsProjectLocked() returns bool value and get its real time value for project by contacting the server.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer = tiaPortal.ProjectServers.First(a =>
a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
LockStateProvider lockStateProvider =
projectServer.GetLockStateProvider(serverProjectInfo[0]);
bool isServerProjectLocked = lockStateProvider.IsProjectLocked();
```

5.15.18 Getting lock owner

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the GetLockOwner() to get information about the lock owner of server project. The GetLockOwner() returns string and returns empty string if the project is not locked.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer =
tiaPortal.ProjectServers.First(a=>a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
LockStateProvider lockStateProvider =
projectServer.GetLockStateProvider(serverProjectInfo[0]);
string lockOwner= lockStateProvider.GetLockOwner();
```

See also

Opening a project (Page 128)

5.15.19 Checking session UptoDate

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the IsUptoDate() to check if the session for multiuser is up-to-date or not. The IsUptoDate() returns the bool value.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
string aliasName = "ProjectServer1";
ProjectServer projectServer = tiaPortal.ProjectServers.First(a =>
a.ServerName.Equals(aliasName));
IEnumerable<ServerProjectInfo> serverProjectInfo = projectServer.GetServerProjects();
string localSessionName = "testLocalSessionName";
DirectoryInfo directoryInfo = new DirectoryInfo("C:\\Sessions");
LocalSessionInfo localSessionInfo =
projectServer.CreateLocalSession(serverProjectInfo.First(), localSessionName,
directoryInfo,
SessionCreationMode.Exclusive);
LocalSession multiuserLocalSession =
tiaPortal.LocalSessions.Open(localSessionInfo.ProjectFileInfo);
bool isUptoDate = multiuserLocalSession.IsUptoDate();
```

Note

In the above progam code the sessionmode for project server can either be multiuser or exclusive.

5.15.20 Getting markings from local session

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to retrieve all the markings (conflicted, non-conflicted) with in the local session.

The TIA Portal Openness can be used to retrieve the online/real-time information from the multiuser server.

The following method can be accessed using TIA Portal Openness:

Method name	Return Type
GetMarkings	Markings

The following property name can be accessed using TIA Portal Openness:

Custom Type	Property name	Return Type
Markings	AllMarkings	MarkingComposition
Markings	ConflictedMarkings	MarkingComposition
Marking	MarkedObject	IEngineeringObject
Marking	MarkState	MarkState

The TIA Portal Openness API does not return markings on objects which got deleted as openness framework cannot represent deleted objects.

Note

Conflicted markings means markings that are marked by you and others or marked by you and outdated or marked by you,marked by others and outdated.

Program code

```
//Open a local session or a server project
//All the non deleted object markings from the session will be retrieved.
Markings markings = localSession.MarkingService.GetMarkings();
MarkingComposition allMarkings = markings.AllMarkings;
foreach (Marking marking in allMarkings)
{
IEngineeringObject markedObject = marking.MarkedObject;
MarkState markState = marking.MarkState;
}
//All the non deleted marked objects which are conflicted.
MarkingComposition conflictedMarkings = markings.ConflictedMarkings;
```

5.16 Functions for accessing data of the user management

5.16.1 Functions for accessing user, roles and function rights

5.16.1.1 Protecting Project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness application to protect the TIA Portal project. The API will protect the project and create a default admin.

The TIA Portal will fail to protect the project under given condition:

- Protecting a previously protected project
- Meta configuration do not exists for at-least one UmacDevice in the project
- Validation fails for provided parameters

The EngineeringTargetInvocationException will be thrown If the protectProject() fails to call.

Program code

```
private static SecureString GetSecureString(string value)
{
SecureString secureStr = new SecureString();
if (!string.IsNullOrEmpty(value))
{
for (int i = 0; i < value.Length; i++)
{
secureStr.AppendChar(value[i]);
}
}
return secureStr;
}
private static void Main(string[] args)
{
TiaPortal tiaPortal = new TiaPortal();
Project tiaProject = tiaPortal.Projects[0];
tiaProject.ProtectProject("AdminUser", GetSecureSting("Admin123"));// see GetSecureString
function
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.1.2 Engineering Function Rights

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness application to retrieve the following information related to Engineering function rights:

- Retrieve all Engineering Function Rights present in a TIA Portal Project
- Retrieve a particular Engineering Functions right by Identifier

For Engineering Function Right, Name property value will be based on user interface language. But Identifier property value will be always constant irrespective of user interface language.

For "Find" API Identier property value should be used.

Program code

```
private static SecureString GetSecureString(string value)
{
SecureString secureStr = new SecureString();
if (!string.IsNullOrEmpty(value))
{
for (int i = 0; i < value.Length; i++)
{
secureStr.AppendChar(value[i]);
}
}
return secureStr;
}
```

```
Project tiaProject = ...;
tiaProject.protectProject("Admin123", ToSecureSting("Admin123"));// see fucntion
GetSecureString
UmacConfigurator UmacConfiguratorService = tiaProject.GetService<UmacConfigurator>();
//Retrieving Engineering function Rights. Returned list will be empty if called in non-
protected mode
var engineeringFunctionRights = UmacConfiguratorService.EngineeringFunctionRights;
string engineeringFunctionRightIdentifier;
string engineeringFunctionRightName;
//Iterating through Engineering function Rights
foreach((var engineeringFunctionRight in engineeringFunctionRights)
{
engineeringFunctionRightName = engineeringFunctionRight.Name;
engineeringFunctionRightIdentifier = engineeringFunctionRight.Identifier;
}
//Find an Engineering Function Right
var foundEngineeringFunctionRight =
engineeringFunctionRights.Find("engineeringFunctionRightIdentifier");
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.1.3 Device Function Rights

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

The device function rights are of two types in TIA Portal:

- System device function rights
- Custom device function rights

You can use the TIA Portal Openness application to retrieve available device function rights:

- Retrieve all device function rights associated to a system and custom device instance
- Retrieve all custom device function rights in the project
- Create a custom device function rights in the project
- Retrieve a custom device function rights by Identifier

Note

The TIA Portal Openness will throw EngineeringTargetInvocationException for creating a custom device function right.

The following scenarios where creating a custom device function rights will fail:

- Maximum custom device function rights limit is reached and no more new custom device function rights possible
- Validation fails for given parameters (like special characters or length)

For System Device Function Right, the Name property value will be defined based on TIA Portal interface language. The Identifier property value will always constant irrespective of the language set in the TIA Portal user interface.

For "Find" API Identifier property value should be used.

Program code

```
Device device = ...;
Project project = ...;
var umacDevice = device.GetService<UmacDevice>();
// Get available system and custom device function rights in the project
var deviceFunctionRightAssociation = umacDevice.AvailableDeviceFunctionRights;
string name;
string identifier;
// Iterating through device function rights
foreach(var deviceFunctionRight in deviceFunctionRightAssociation)
{
name = deviceFunctionRight.Name;
identifier = deviceFunctionRight.Identifier;
}
// Retrieve the UmacConfiguratorService from project
UmacConfigurator umacConfiguratorService = project.GetService<UmacConfigurator>();
// Get all custom device function rights in the project
var CustomDeviceFunctionRightComposition =
umacConfiguratorService.AvailableCustomDeviceFunctionRights;
//create a custom device function right
var CustomDeviceFunctionRights =
customDeviceFunctionRightComposition.Create("RuntimeRight1", "General", "user defined
runtime right");
// Name and Identifier value for Custom Device Function Right will be the same.
// Iterating through custom device function rights
foreach(var customDeviceFunctionRight in CustomDeviceFunctionRights)
{
name = customDeviceFunctionRight.Name;
identifier = customDeviceFunctionRight.Identifier;
}
//Find a custom device function right
customDeviceFunctionRight = customDeviceFunctionRightComposition.Find("identifier");
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.1.4 System Roles

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can perform the following tasks with custom role and device function rights while using TIA Portal Openness:

- Retrieve all system roles in a project
- Finding a particular system role in a project
- Retrieve all assigned engineering rights to a given system role
- Retrieve all assigned system device function rights for system role

For System Roles, The value for Name property will be defined based on TIA Portal user interface language. The value for Identifier property will be always constant irrespective of language in TIA Portal user interface .

For "Find" API Identifier property value should be used.

Program code

```
Project project = ...;
Device device = ...;
UmacConfigurator umacConfiguratorService = project.GetService<UmacConfigurator>();
var umacDevice = device.GetService<UmacDevice>();
//Retrieve all system role
var systemRoles = umacConfiguratorService.SystemRoles;
//Iterate through the system roles
foreach (var sr in systemRoles)
{
string systemRoleName = sr.Name;
}
// Find a system role
SystemRole systemRole = systemRoles.Find("SystemRole_01");
//Retrieve assigned System Device Function Rights from system role
IList<SystemDeviceFunctionRight> assignedSystemDeviceFunctionRights
=(IList<SystemDeviceFunctionRight>)systemRole.
GetAssignedSystemDeviceFunctionRights(umacDevice);
//Retrieve assigned Engineering Function Rights from system role
var engineeringFunctionRights = systemRole.AssignedEngineeringRights;
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.1.5 Custom Roles

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

The custom roles are user created roles which are dynamically created during the course of project life time. You can perform the following tasks with custom role while using TIA Portal Openness:

- Retrieve all Custom roles in a project
- Creating a custom role in a project
- Finding a particular custom role in a project
- Editing the attributes(name and comment) of a custom role
- Deleting a custom role in a project
- Assigning/ Un assigning an EFR to a custom role
- Assigning/ Un assigning an DFR to a custom role via a device instance
- Retrieving Assigned Engineering Function Rights to a Custom Role
- Retrieving Assigned Device Function Rights to a Custom Role

For Custom Role, Identifier property value will be always constant irrespective of user interface language. For "Find" API Identifier property value should be used.

Program code

```
TiaPortal tiaPortal = new TiaPortal();
Project tiaProject = tiaPortal.Projects[0];
UmacConfigurator UmacConfiguratorService = tiaProject.GetService<UmacConfigurator>();
// Retrieve custom roles in project
var customRoles = UmacConfiguratorService.CustomRoles;
// Create a custom role
CustomRole newCustomRole = customRoles.Create("Role_1", "some comment");
// Iterate through custom roles & edit it
foreach (CustomRole customRole in customRoles)
{
if (newCustomRole.Name == "CustomRoleNew")
{
customRole.Comment = "CustomRoleNew is allowed to ...";
//..
}
}
// Find a custom role
newCustomRole = customRoles.Find("CustomRole1");
// Edit a custom role
 newCustomRole.Comment = "some new comment";
// Retrieve Engineering Function Right. This list will be empty if called in non-protected
mode.
var engineeringFunctionRights = UmacConfiguratorService.EngineeringFunctionRights;
// Find an Engineering Function Right
EngineeringFunctionRight engineeringFunctionRight =
engineeringFunctionRights.Find("EFR_01");
// Assign a EFR to Custom Role
customRole.AssignedEngineeringRights.Add(engineeringFunctionRight);
// Retrieve assigned Engineering Function Rights on a custom role
var engineeringFunctionRights = customRole.AssignedEngineeringRights;
// Remove a EFR from Custom Role
customRole.AssignedEngineeringRights.Remove(engineeringFunctionRight);
// Get the umac device service from openness Device. If service is invoked on non-umac
device then value will be null
Device device = ...;
var umacDevice = device.GetService<UmacDevice>();
var deviceFunctionRightsAssociation = umacDevice.AvailableDeviceFunctionRights;
var customDeviceFunctionRight = deviceFunctionRightsAssociation.Find("User Right1");
var systemDeviceFunctionRight = deviceFunctionRightsAssociation.Find("SystemDFR_01");
// Assign a custom device function right to custom role
customRole.AssignDeviceFunctionRight(umacDevice, customDeviceFunctionRight);
//Assign a system device function right
customRole.AssignDeviceFunctionRight(umacDevice, systemDeviceFunctionRight);
//Retrieve assigned Device function Rights
var assignedDeviceFunctionRights = customRole.GetAssignedDeviceFunctionRights(umacDevice);
// Unassign a custom device function right from custom role
customRole.UnAssignDeviceFunctionRight(umacDevice, customDeviceFunctionRight);
//Unassign a system device function right
customRole.UnAssignDeviceFunctionRight(umacDevice, systemDeviceFunctionRight);
// Delete a Custom Role
customRole.Delete();
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.1.6 Project Users

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

The Project users are users created locally for a project. When a Project is protected by default an Project user is created and is given ES Administrative role.

You can perform the following tasks with project users while using TIA Portal Openness:

- Retrieve all Project users in a project
- Create a Project user in a project
- Find a Project user by name in a project
- Get the Assigned Roles for a Project user
- Add/ Remove a role to a Project user
- Activate/Deactivate project user
- Indicates whether the Project user is Active

Program code

```
private static SecureString GetSecureString(string value)
{
SecureString secureStr = new SecureString();
if (!string.IsNullOrEmpty(value))
{
for (int i = 0; i < value.Length; i++)
{
secureStr.AppendChar(value[i]);
}
}
return secureStr;
}
private static void protectProject()
{
TiaPortal tiaPortal = new TiaPortal();
Project tiaProject = tiaPortal.Projects[0];
tiaProject.ProtectProject("AdminUser", GetSecureString("Admin123"));//see fucntion
GetSecureString
}
private static void projectUser()
{
Project project = ...;
UmacConfigurator UmacConfiguratorService = project.GetService<UmacConfigurator>();
//Retrieve custom roles in project
var customRoles = UmacConfiguratorService.CustomRoles;
var customRole = customRoles.Find("Role_1");
var systemRole = customRoles.Find("Engineering administrator");
//Get project users
var projectUsers = UmacConfiguratorService.ProjectUsers;
// Find a project user
ProjectUser newProjectUser = projectUsers.Find("AdminUser");
if (newProjectUser == null)
{
// Create a project user
newProjectUser = projectUsers.Create("AdminUser", GetSecureString("AdminUser123#"));
}
// Iterate through existing project users
foreach (ProjectUser projectUser in projectUsers)
{
string projectUserName = projectUser.Name;
// Indicates whether the project user is Active
var isUserActive = projectUser.IsActive;
//Change password of project user
projectUser.SetPassword(GetSecureString("NewPassword@123"));
//Change AuthenticationType of project user
projectUser.ProjectUserAuthenticationType = AuthenticationType.Radius;
// Deactivate project project user
projectUser.Deactivate();
// Add a Custom Role to a deactivated project user
projectUser.Roles.Add(customRole);
// Add a System Role to a deactivated project user
projectUser.Roles.Add(systemRole);
// Remove a Custom Role from a deactivated project user
bool isSuccessfullyRemoved = projectUser.Roles.Remove(customRole);
```

```
// Remove a System Role from a deactivated project user
isSuccessfullyRemoved = projectUser.Roles.Remove(systemRole);
// Activate project project user
projectUser.Activate();
// Delete the project user instance
projectUser.Delete();
}
}
```

See also

Connecting to the TIA Portal (Page 82)

5.16.2 Functions for UMAC Global Users and UMC Server

5.16.2.1 Authentication to connect to a UMC Server

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the UmcServerConfigurator service on UmcServer property to get the list of UMC Servers. The authentication event is invoked on the UMC server to connect to the server.

The UmcServerConfigurator service is available for both protected and unprotected project modes. The authentication event will not be triggered while importing UMC users or groups from UMC server for UMC admin logged in as an user.

The authentication event will trigger only when the logged user is other that UMC user with UMC View Right at UMC Server. For example, project user, UMC user who is not having UMC View Right at UMC Server.

With TIA Portal Openness V17, only one UMC Server is available and preconfigured.

Program code

```
private static SecureString GetSecureString(string value)
{
SecureString secureStr = new SecureString();
if (!string.IsNullOrEmpty(value))
{
for (int i = 0; i < value.Length; i++)
{
secureStr.AppendChar(value[i]);
}
}
return secureStr;
}
```

```
Project project = ...;
var umcServerConfigurator = project.GetService<UmcServerConfigurator>();
UmcServer umcServer = umcServerConfigurator.UmcServer;
umcServer.Authentication += UmcServer_Authentication;
```

The UmcServer_Authentication() have Event args where you can pass the UMC admin name and password.

```
private static void UmcServer_Authentication(object sender, UmcAuthenticationEventArgs e)
{
e.UmcCredentials.Name = "Admin";
Console.WriteLine("Name is set");
e.UmcCredentials.SetPassword(GetSecureString("Admin123"));
Console.WriteLine("Password is set");
}
```

5.16.2.2 Retrieving an UMC User Group from a UMC Server

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to retrieve an UMC User Group from a UMC Server. The GetUserGroupByName ( ) method is available at UMC server level to get UMC user group of the connected UMC server.

When the GetUserGroupByName ( ) method is triggered, the event call back triggers for the UMC Server authentication. Only UMC user having "UMC View" right will be available to retrieve an UMC user group.

Program code

```
Project project = ...;
var umcServerConfigurator = project.GetService<UmcServerConfigurator>();
UmcServer umcServer = umcServerConfigurator.UmcServer;
umcServer.Authentication += UmcServer_Authentication;
string umcUserGroupName = "xxxxxx";
Siemens.Engineering.Umac.UmcUserGroupInfo umcUserGroupsFromUmcServer =
umcServer.GetUserGroupByName(umcUserGroupName);
```

5.16.2.3 Retrieving an UMC User from a UMC Server

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to retrieve UMC user from UMC server. The GetUserByName( ) on the UMC server level can be used to retrieve the corresponding UMC user from UMC server by the UMC username.

When the GetUserByName( ) method is triggered, the event call back triggers for the UMC Server Authentication. Only Umc user with UMC View Right will be able to fetch a UMC user from the UMC server.

Program code

```
Project project = ...;
var umcServerConfigurator = project.GetService<UmcServerConfigurator>();
UmcServer umcServer = umcServerConfigurator.UmcServer;
umcServer.Authentication += UmcServer_Authentication;
string umcUserName = "xxxxxx";
Siemens.Engineering.Umac.UmcUserInfo umcUsersFromUmcServer =
umcServer.GetUserByName(umcUserName);
```

Note

All UMC exceptions are handled as Siemens.Engineering.TargetInvocation Exception in TIA Portal Openness.

5.16.2.4 Adding an UMC User Group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to add an UMC user group to TIA Portal project from an UMC server.

Program code

```
Project project = ...;
var umcServerConfigurator = project.GetService<UmcServerConfigurator>();
var umacConfigurator = project.GetService<UmacConfigurator>();
UmcServer umcServer = umcServerConfigurator.UmcServer;
umcServer.Authentication += UmcServer_Authentication;
string umcUserGroupName = "xxxxxx";
Siemens.Engineering.Umac.UmcUserGroupInfo umcUserGroupInfo =
umcServer.GetUserGroupByName(umcUserGroupName);
//Get the UmcUserGroupComposition.
UmcUserGroupComposition umcUserGroupComposition = umacConfigurator.UmcUserGroups;
//Create the selected UMC UserGroup in to the TIA Portal project
UmcUserGroup umcUserGroup = umcUserGroupComposition.Create(umcUserGroupInfo);
```

5.16.2.5 Getting all UMC groups in TIA Portal

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to get all UMC user groups added to TIA Portal projects.

When there are no UMC user groups, the TIA Portal Openness application will return empty collection.

Program code

```
//Get UmcUserGroups from the Project.
UmcUserGroupComposition umcUserGroupComposition = umacConfigurator.UmcUserGroups;
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.2.6 Adding an UMC User to TIA Portal

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to add an UMC user to the TIA Portal project after the UMC users are retrieved from UMC server.

Program code

```
Project project = ...;
var umcServerConfigurator = project.GetService<UmcServerConfigurator>();
var umacConfigurator = project.GetService<UmacConfigurator>();
UmcServer umcServer = umcServerConfigurator.UmcServer;
umcServer.Authentication += UmcServer_Authentication;
string umcUserName ="xxx";
Siemens.Engineering.Umac.UmcUserInfo umcUserInfo = umcServer.GetUserByName(umcUserName);
//Gets the UmcUserComposition
UmcUserComposition umcUserComposition = umacConfigurator.UmcUsers;
//Create the selected UMC user into the TIA Portal project
UmcUser umcUser = umcUserComposition.Create(umcUserInfo);
```

See also

Opening a project (Page 128)

5.16.2.7 Getting all UMC Users in TIA Portal project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to get all the UMC users in TIA Portal.

When there are no UMC users, the TIA Portal Openness application will return empty collection.

Program code

```
var umacConfigurator = project.GetService<UmacConfigurator>();
UmcUserComposition umcUserComposition = umacConfigurator.UmcUsers;
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.2.8 Finding an UMC User added in TIA Portal

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to find corresponding UMC user from the TIA Portal project.

Program code

```
string UmcUserName ="XXXX";
UmcUserComposition umcUsersList = umacConfigurator.UmcUsers;
UmcUser umcUser = umcUsersList.Find(UmcUserName);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.2.9 Finding an UMC User group added in TIA Portal project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to find corresponding UMC User group from the TIA Portal project.

Program code

```
//Get UmcUserGroups from the Project.
string UmcUserGroupName ="XXXX";
UmcUserGroupComposition umcUserGroupList = umacConfigurator.UmcUserGroups;
UmcUserGroup umcUserGroup = umcUserGroupList.Find(UmcUserGroupName);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.2.10 Assigning role to UMC User

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to add the corresponding role to the UMC user added to TIA Portal project.

Program code

```
//Assign the System Role.
UmacConfigurator umacConfigurationBuilder = ...;
string s_SystemRoleName = "";
SystemRole engineeringSystemRole =
umacConfigurationBuilder.SystemRoles.Find(s_SystemRoleName);
UmcUser umcUser = ...;
umcUser.Roles.Add(engineeringSystemRole);
//Assign the Custom Role.
UmacConfigurator umacConfigurator = ...;
CustomRole customRole = umacConfigurator.CustomRoles.Create("CustomRole");
```

See also

Opening a project (Page 128)

5.16.2.11 Getting the list of assigned roles of UMC User and UMC User Group.

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to get the list of assigned roles of UM user and UMC user group.

Program code

```
//Get assigned roles from UmcUserGroup.
string UmcUserGroupName ="XXXX";
UmcUserGroupComposition umcUserGroupList = umacConfigurationBuilder.UmcUserGroups;
UmcUserGroup umcUserGroup = umcUserGroupList.Find(UmcUserGroupName);
RoleAssociation assignedRolesInUmcUserGroup = umcUserGroup.Roles;
//Get assigned roles from UmcUser.
string UmcUserName ="XXXX";
UmcUserComposition umcUserList = umacConfigurationBuilder.UmcUsers;
UmcUser umcUser = umcUserList.Find(UmcUserGroupName);
RoleAssociation assignedRolesInUmcUser = umcUser.Roles;
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.2.12 Assigning role to UMC User Group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to add the corresponding role to the UMC user group added to TIA Portal project.

Program code

```
//Assign the System Role.
UmacConfigurator umacConfigurationBuilder = ...;
string s_SystemRoleName = "";
SystemRole engineeringSystemRole =
umacConfigurationBuilder.SystemRoles.Find(s_SystemRoleName);
UmcUserGroup umcUserGroup = ...;
umcUserGroup.Roles.Add(engineeringSystemRole);
//Assign the Custom Role
UmacConfigurator umacConfigurator = ...;
CustomRole customRole = umacConfigurator.CustomRoles.Create("CustomRole");
umcUserGroup.Roles.Add(customRole);
```

See also

Opening a project (Page 128)

5.16.2.13 Removing role from UMC User and UMC User Group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Application to unassign the corresponding role from the UMC user and UMC user group added to TIA Portal project.

Program code

Modify the following program code to remove role from the UMC user added to TIA Portal project.

```
//Remove the System Role.
bool isSuccessfullyRemoved = umcUser.Roles.Remove(engineeringSystemRole);
//Remove the Custom Role
isSuccessfullyRemoved = umcUser.Roles.Remove(customRole);
```

Modify the following program code to remove role from the UMC user group added to TIA Portal project.

```
//Un assign the System Role.
bool isSuccessfullyRemoved = umcUserGroup.Roles.Remove(engineeringSystemRole);
//Un assign the Custom Role.
isSuccessfullyRemoved = umcUserGroup.Roles.Remove(customRole);
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.16.2.14 Activating/Deactivating UMC User and UMC User Group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal
- A project is open
  See Opening a project

Application

You can use the TIA Portal Opennness to activate / deactivate UMC user and UMC user group.

Program code

```
//Activate UMC User
UmcUser umcUserToActivate = umacConfigurator.UmcUsers.Find("UserNameToActivate");
umcUserToActivate.Activate();
//Deactivate UMC User
UmcUser umcUserToDeactivate = umacConfigurator.UmcUsers.Find("UserNameToDeactivate");
umcUserToDeactivate.Deactivate();
//Activate UMC User Group
UmcUserGroup umcUserGroupToActivate =
umacConfigurator.UmcUserGroups.Find("UserGroupNameToActivate");
umcUserGroupToActivate.Activate();
//Deactivate UMC User Group
UmcUserGroup umcUserGroupToDeactivate =
umacConfigurator.UmcUserGroups.Find("UserGroupNameToDeactivate");
umcUserGroupToDeactivate.Deactivate();
```

5.16.2.15 Checking state for UMC User and UMC User Group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to retrieve activation state of UMC user and UMC user group.

Program code

```
//Indicates whether the User is Active
UmcUser umcUserToActivate = umacConfigurator.UmcUsers.Find("UserNameToActivate");
var isUserActive = umcUserToActivate.IsActive;
//Indicates whether the Group is Active
UmcUserGroup umcUserGroupToDeactivate =
umacConfigurator.UmcUsersGroup.Find("UserGroupNameToActivate");
var isGroupActive = umcUserGroupToDeactivate.IsActive;
```

See also

Opening a project (Page 128)

5.16.2.16 Deleting UMC User and UMC User Group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to delete UMC user and UMC User Group from TIA Portal project.

Program code

```
//Delete user from TIA Portal project.
UmcUser umcUserToDelete = umacConfigurator.UmcUsers.Find("UserNameToDelete");
umcUserToDelete.Delete();
//Delete of user group from TIA Portal project.
UmcUserGroup umcUserGroupToDelete =
umacConfigurator.UmcUserGroups.Find("GroupNameToDelete");
umcUserGroupToDelete.Delete();
```

See also

Connecting to the TIA Portal (Page 82)

5.16.2.17 Synchronizing UMC user

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can use the Synchronize() in the TIA Portal Openness to synchronize the UMC data. The Synchronize() retrieves the UMC user and UMC user group in the project if the UMC user and UMC user group present in the project is configured in the domain project.

Program code

```
private void SynchronizeUMCUser()
{
m_UmcServerConfigurator = project.GetService<UmcServerConfigurator>();
//Synchronize UMC Data
m_UmcServerConfigurator.Synchronize();
}
```

5.16.2.18 Checking UMC data consistency

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Application

You can use the CheckConsistency() in the TIA Portal Openness to check the consistency of UMC data. The CheckConsistency() returns true if the UMC data in the project matches the configured server.

Program code

```
private void CheckUMCData()
{
m_UmcServerConfigurator = project.GetService<UmcServerConfigurator>();
//Check consistency of UMC Data
m_UmcServerConfigurator.CheckConsistency();
}
```

5.16.2.19 Setting runtime session timeout

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to enable or disable the runtime timeout for an user in the UMAC editor.

Method and Parameter

Method name	Description
ActivateRuntimeSessionTimeout()	Activate the runtime session timeout for an user in the UMAC editor
DeactivateRuntimeSessionTimeout()	Deactivate the runtime session timeout session for an user in the UMAC editor

Property name	Data type	Description
IsRuntimeSessionTimeoutActive	Boolean	Gets the current activation state of runtime session timeout property. The IsRuntimeSessionTimeoutActive returns true if it is active. Otherwise it returns false for deactivate.

Program code

```
private void SettingRuntimeSession()
{
var umacConfigurator = Project.GetService<UmacConfigurator>();
var projectUsers = umacConfigurator.ProjectUsers;
var createdProjectUser = projectUsers.Create("ProjectUser1",
"ProjectUser123".ToSecureString());
//Gets the current activation state of runtime session timeout property, true if active and
false if deactivated
createdProjectUser.IsRuntimeSessionTimeoutActive;
//deactivates the runtime session timeout property
//cannot deactivate if already deactivated
createdProjectUser.DeactivateRuntimeSessionTimeout();
//Activates the runtime session timeout property
//cannot activate if already activated
createdProjectUser.ActivateRuntimeSessionTimeout();
}
```

5.16.2.20 Setting password policies for UMAC

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the passwordPolicyConfigurator() in the TIA Portal Openness to set the password policies for Runtime and Engineering password settings

Property

Property Name	Data type	Description	Access
IncludesLowerCaseAndUpperCaseCharacters	Boolean	Sets or gets the value of includesLowerCaseAndUpperCaseCharacters	Read/Write
MinimumNumericCharacterLength	Short	Sets or gets the value of minimumNumericCharacterLength in password	Read/Write
MinimumLength	Short	Sets or gets the value of minimumLength of the password	Read/Write
MinimumSpecialCharacterLength	Short	Sets or gets the value of minimumSpecialCharacterLength in password	Read/Write
EnablePasswordAging	Short	Sets or gets the value of enablePasswordAging	Read/Write
MinimumUserPasswordsBlockedForReuse	Short	Sets or gets the value of minimumUserPasswordBlockedForReuse	Read/Write
PasswordValidityPrewarningTime	Short	Sets or gets the value of passwordValidityPrewarningTime in days which is out of range	Read/Write
PasswordValidity	Short	Sets or gets the value of passwordValidity in days which is out of range	Read/Write

Program code

```
private void PasswordPolicySetting()
{
var passwordPolicyConfigurator = tiaProject.GetService<PasswordPolicyConfigurator>();
//Sets the value of includesLowerCaseAndUpperCaseCharacters
var includesLowerCaseAndUpperCaseCharacters = true;
passwordPolicyConfigurator.IncludesLowerCaseAndUpperCaseCharacters =
includesLowerCaseAndUpperCaseCharacters;
//Gets the value of includesLowerCaseAndUpperCaseCharacters
includesLowerCaseAndUpperCaseCharacters =
passwordPolicyConfigurator.IncludesLowerCaseAndUpperCaseCharacters;
// Sets the value of minimumNumericCharacterLength in password
var minimumNumericCharacterLength = 2;
passwordPolicyConfigurator.MinimumNumericCharacterLength = minimumNumericCharacterLength;
// Gets the value of minimumNumericCharacterLength in password
minimumNumericCharacterLength = passwordPolicyConfigurator.MinimumNumericCharacterLength;
//Sets and gets the value of minimumLength of the password
var minimumLength = 11;
passwordPolicyConfigurator.MinimumLength = minimumLength;
//Gets the value of minimumLength of the password
minimumLength = passwordPolicyConfigurator.MinimumLength;
//Sets the value of minimumSpecialCharacterLength in password
var minimumSpecialCharacterLength = 1;
passwordPolicyConfigurator.MinimumSpecialCharacterLength = minimumSpecialCharacterLength;
//Gets the value of minimumSpecialCharacterLength in password
minimumSpecialCharacterLength = passwordPolicyConfigurator.MinimumSpecialCharacterLength;
//Sets the value of enablePasswordAging
var enablePasswordAging = true;
passwordPolicyConfigurator.EnablePasswordAging = enablePasswordAgingrue;
//Gets the value of enablePasswordAging
enablePasswordAging = passwordPolicyConfigurator.EnablePasswordAging;
//Sets the value of passwordValidity in days
var passwordValidity = 5;
passwordPolicyConfigurator.PasswordValidity = passwordValidity;
//Gets the value of passwordValidity in days
passwordValidity = passwordPolicyConfigurator.PasswordValidity;
//Sets the value of passwordValidityPrewarningTime in days
var passwordValidityPrewarningTime = 1;
passwordPolicyConfigurator.PasswordValidityPrewarningTime =
passwordValidityPrewarningTime;
//Gets the value of passwordValidityPrewarningTime in days
passwordValidityPrewarningTime =
passwordPolicyConfigurator.PasswordValidityPrewarningTime;
// Sets the value of minimumUserPasswordsBloackedForReuse
var minimumUserPasswordsBloackedForReuse = 2;
passwordPolicyConfigurator.MinimumUserPasswordsBlockedForReuse =
minimumUserPasswordsBloackedForReuse;
// Gets the value of minimumUserPasswordsBloackedForReuse
minimumUserPasswordsBloackedForReuse =
passwordPolicyConfigurator.MinimumUserPasswordsBlockedForReuse;
//Sets the value of minimumLength of the password which is out of range
var minimumLength = -11;
try
{
passwordPolicyConfigurator.MinimumLength = minimumLength;
}
```

```
catch(Siemens.Engineering.PasswordPolicySettingsException exception)
{
Console.Writeline(exception.Message);
}
// Sets the value of minimumNumericCharacterLength in password which is out of range
var minimumNumericCharacterLength = -2;
try
{
passwordPolicyConfigurator.MinimumNumericCharacterLength = minimumNumericCharacterLength;
}
catch(Siemens.Engineering.PasswordPolicySettingsException exception)
{
Console.Writeline(exception.Message);
}
//Sets the value of minimumSpecialCharacterLength in password which is out of range
var minimumSpecialCharacterLength = -1;
try
{
passwordPolicyConfigurator.MinimumSpecialCharacterLength = minimumSpecialCharacterLength;
}
catch(Siemens.Engineering.PasswordPolicySettingsException exception)
{
Console.Writeline(exception.Message);
}
//Sets the value of passwordValidity in days which is out of range
var passwordValidity = -5;
try
{
passwordPolicyConfigurator.PasswordValidity=passwordValidity;
}
catch(Siemens.Engineering.PasswordPolicySettingsException exception)
{
Console.Writeline(exception.Message);
}
//Sets the value of passwordValidityPrewarningTime in days which is out of range
var passwordValidityPrewarningTime = -1;
try
{
passwordPolicyConfigurator.PasswordValidityPrewarningTime =
passwordValidityPrewarningTime;
}
catch(Siemens.Engineering.PasswordPolicySettingsException exception)
{
Console.Writeline(exception.Message);
}
// Sets the value of minimumUserPasswordsBloackedForReuse which is out of range
var minimumUserPasswordsBloackedForReuse = -2;
try
{
passwordPolicyConfigurator.MinimumUserPasswordsBlockedForReuse=minimumUserPasswordsBloacke
dForReuse;
}
catch(Siemens.Engineering.PasswordPolicySettingsException exception)
{
Console.Writeline(exception.Message);
}
```

```
}
```

5.16.2.21 Accessing Anonymous user in protected project

Requirement

- The TIA Portal Openness is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

Anonymous user does not require password to login into protected project. The Anonymous user is created by default in deactivate state when a TIA Portal project is protected. There is only single anonymous user created in a protected project.

You can use the TIA Portal Openness to manage the following functionalities of anonymous user:

- Activating anonymous user
- Deactivating anonymous user
- Retrieving anonymous user in protected project
- Adding and removing system role and custom role to anonymous user

Program code

```
private void ManagingAnonymousUser()
{
UmacConfigurator umacConfiguratorService = project.GetService<UmacConfigurator>();
//Retrieve anonymous user. If not activated user will be nullvar anonymousUser =
umacConfiguratorService.AnonymousUser;
/activate/deactivate anonymous user
umacConfiguratorService.ActivateAnonymousUser()
//Retrieve anonymous user. If not activated user will be null
anonymousUser = umacConfiguratorService.AnonymousUser;
// Add a Custom or System Role to a Anonymous User
anonymousUser.Roles.Add(customRoles.Find("CustomRole_01"));
anonymousUser.Roles.Add(systemRoles.Find("SystemRole_01"));
// Remove a Custom or System Role to a Anonymous User
bool isSuccessfullyRemoved = anonymousUser.Roles.Remove(customRoles.Find("CustomRole_01"));
isSuccessfullyRemoved  = anonymousUser.Roles.Remove(systemRoles.Find("SystemRole_01"));
umacConfiguratorService.DeactivateAnonymousUser();
}
```

5.16.2.22 Complete sample code

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Program code

Below TIA Portal Openness script describes about the complete workflow of Openness APIs for UMAC global users and UMC server.

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
using Siemens.Engineering.Umac;
namespace VollständigerBeispielcode
{
internal class Program
{
static void Main(string[] args)
{
TiaPortal tiaPortal = new TiaPortal();
//Registers for TIA Portal authentication for protected project opening
tiaPortal.Authentication += TiaPortal_Authentication;
Project umacProject = tiaPortal.Projects.Open(new FileInfo("ProtectedProject_Path"));
var umcServerConfigurator = umacProject.GetService<UmcServerConfigurator>();
// Getting UMC servers.
UmcServer umcServer = umcServerConfigurator.UmcServer;
// Register for UMC Server Authentication
umcServer.Authentication += UmcServer_Authentication;
var umacConfigurator = umacProject.GetService<UmacConfigurator>();
// Use case :Import User Group, Add User Group to Tia portal project , Assign & Unassign
Role to UserGroup
// Delete the User group from Tia portal project.
// Retrieve an UMC User Group from UMC Server
const string UmcUserGroupName = "xxxxx";
UmcUserGroupInfo umcUserGroupInfo = umcServer.GetUserGroupByName(UmcUserGroupName);
// Add an UMC User Group into TIA portal Project.
UmcUserGroupComposition umcUserGroupComposition = umacConfigurator.UmcUserGroups;
UmcUserGroup umcUserGroup = umcUserGroupComposition.Create(umcUserGroupInfo);
// Assign and Unassign roles to User Group.
//Assign the System Role.
const string SystemRoleName = "Engineering administrator";
SystemRole engineeringAdministratorSystemRole =
umacConfigurator.SystemRoles.Find(SystemRoleName);
umcUserGroup.Roles.Add(engineeringAdministratorSystemRole);
```

```
//Assign the Custom Role.
CustomRole customRole = umacConfigurator.CustomRoles.Create("CustomRole");
umcUserGroup.Roles.Add(customRole);
//Un assign the System Role.
umcUserGroup.Roles.Remove(engineeringAdministratorSystemRole);
//Un assign the Custom Role.
umcUserGroup.Roles.Remove(customRole);
//Delete the UMC user group in Tia Portal project.
UmcUserGroup umcUserGroupToDelete = umacConfigurator.UmcUserGroups.Find(UmcUserGroupName);
umcUserGroupToDelete.Delete();
// Use case :Import UMC User , Add UMC User to Tia portal project , Assign & Unassign Roles
to UMC User,
// Delete the UMC User from Tia portal project.
// Import UMC User from UMC Server
const string UmcUserName = "xxxxx";
UmcUserInfo umcUserInfo = umcServer.GetUserByName(UmcUserName);
//Add UMC User to Tia Portal project
UmcUserComposition umcUserComposition = umacConfigurator.UmcUsers;
UmcUser umcUser = umcUserComposition.Create(umcUserInfo);
// Assign & Unassign Roles to UMC User
//Assign the System Role.
const string SystemRoleNameForUser = "NET Radius";
SystemRole systemRoleForUmcUser = umacConfigurator.SystemRoles.Find(SystemRoleNameForUser);
umcUser.AssignRole(systemRoleForUmcUser); //Needs to be:
umcUser.Roles.Add(systemRoleForUmcUser);
//Assign the Custom Role.
CustomRole customRoleNameForUmcUser =
umacConfigurator.CustomRoles.Create("CustomRoleName");
umcUser.Roles.Add(customRoleNameForUmcUser);
//Un assign the System Role.
umcUser.Roles.Remove(systemRoleForUmcUser);
//Un assign the Custom Role.
umcUser.Roles.Remove(customRoleNameForUmcUser);
//Delete the UMC User from Tia Portal project
UmcUser umcUserToDelete = umacConfigurator.UmcUsers.Find(UmcUserName);
umcUserToDelete.Delete();
}
private static void TiaPortal_Authentication(object sender, AuthenticationEventArgs e)
{
e.AuthenticationTypeProvider = AuthenticationTypeProvider.Credentials;
e.UmacCredentials.Type = UmacUserType.Project;
e.UmacCredentials.Name = "Admin";
e.UmacCredentials.SetPassword(GetSecureString("Admin123"));
}
private static void UmcServer_Authentication(object sender, UmcAuthenticationEventArgs e)
{
e.UmcCredentials.Name = "Admin";
e.UmcCredentials.SetPassword(GetSecureString("Admin123"));
}
private static SecureString GetSecureString(string password)
{
SecureString secureString = new SecureString();
foreach (char c in password)
{
secureString.AppendChar(c);
}
```

```
return secureString;
}
}
}
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.17 Functions on OPC

5.17.1 Configuring OPC UA server secure communication protocol

Requirements

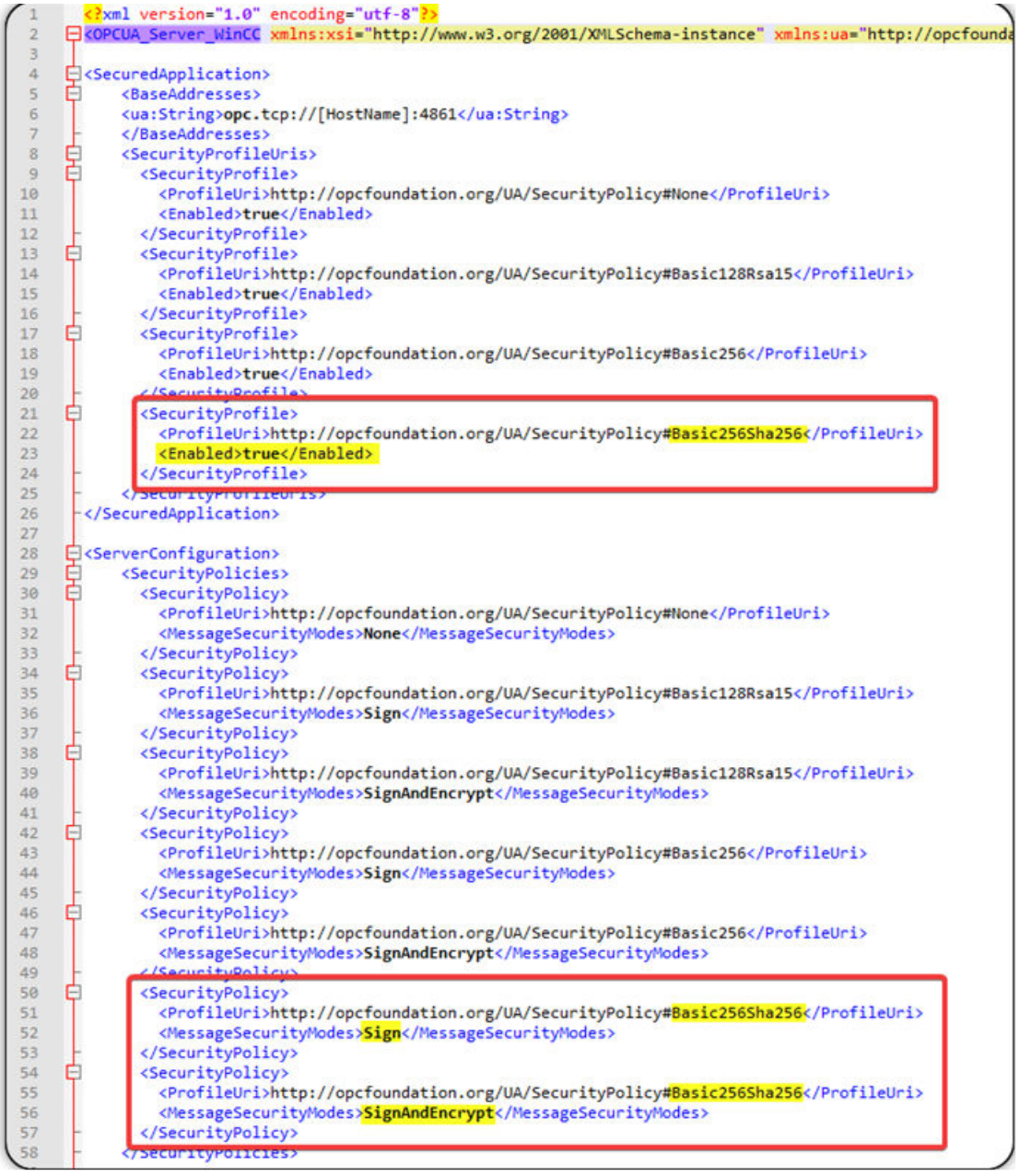
- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness application to configure OPC UA server with security policy "Basic256Sha256" . The security policy Basic256Sha256 needs to be added to the Runtime Settings. RDP needs to compile the properties in the xml configuration file.

The defaults are Enabled, Sign, and Sign and Encrypt.

In the XML file <Project>\OPC\uaserver\OPCUaServerWinCCPro.xml, you need to set the security policy and security policies according to ES device configuration.



```
<?xml version="1.0" encoding="utf-8"?>
<OPCUA_Server_WinCC xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ua="http://opcfounda

<SecuredApplication>
    <BaseAddresses>
    <ua:String>opc.tcp://[HostName]:4861</ua:String>
    </BaseAddresses>
    <SecurityProfileUris>
      <SecurityProfile>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#None</ProfileUri>
        <Enabled>true</Enabled>
      </SecurityProfile>
      <SecurityProfile>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
        <Enabled>true</Enabled>
      </SecurityProfile>
      <SecurityProfile>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
        <Enabled>true</Enabled>
      </SecurityProfile>
      <SecurityProfile>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
        <Enabled>true</Enabled>
      </SecurityProfile>
    </SecurityProfileUris>
</SecuredApplication>

<ServerConfiguration>
    <SecurityPolicies>
      <SecurityPolicy>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#None</ProfileUri>
        <MessageSecurityModes>None</MessageSecurityModes>
      </SecurityPolicy>
      <SecurityPolicy>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
        <MessageSecurityModes>Sign</MessageSecurityModes>
      </SecurityPolicy>
      <SecurityPolicy>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
        <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
      </SecurityPolicy>
      <SecurityPolicy>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
        <MessageSecurityModes>Sign</MessageSecurityModes>
      </SecurityPolicy>
      <SecurityPolicy>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
        <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
      </SecurityPolicy>
      <SecurityPolicy>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
        <MessageSecurityModes>Sign</MessageSecurityModes>
      </SecurityPolicy>
      <SecurityPolicy>
        <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
        <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
      </SecurityPolicy>
    </SecurityPolicies>
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.17.2 Setting OPC UA security policy

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  Opening a project (Page 128)
- OPC UA server is activated

Introduction

You can use the TIA Portal Openness application to set the security policy in OPC UA. You can implement the security policy as a dynamic attribute of type flagged enum: OpcUaSecurityPolicies. The security policy is only available in TIA Portal Openness if the OPC UA server is activated.

If the OPC UA server is deactivated, an EngineeringNotSupportedException will be thrown while trying to access the security policy for any other unavailable attribute.

The below table shows the possible values can be found for security policies:

TIA UI name	Enum entry	Value	Remarks
-	NoneSelected	0	Is equivalent to TIA UI when no checkbox is selected.
No security	OpcUaSecurityPoliciesNone	1	
Basic128Rsa15 - Sign	OpcUaSecurityPolicies128RSAS	2	
Basic128Rsa15 - Sign & Encrypt	OpcUaSecurityPolicies128RSASE	4	
Basic256 - Sign	OpcUaSecurityPolicies256S	8	
Basic256 - Sign & Encrypt	OpcUaSecurityPolicies256SE	16	
Basic256Sha256 - Sign	OpcUaSecurityPolicies256SHAS	32	
Basic256Sha256 - Sign & Encrypt	OpcUaSecurityPolicies256SHASE	64	
Aes128 - Sha256 - RsaOaep - Sign	OpcUaSecurityPolicies256RSAOAEPS	128	
Aes128Sha256 - RsaOaep - Sign & Encrypt	OpcUaSecurityPolicies256RSAOAEPSE	256	
Aes256Sha256RsaPss - Sign	OpcUaSecurityPolicies256RSAPSSS	512	
Aes256Sha256RsaPss - Sign & Encrypt	OpcUaSecurityPolicies256RSAPSSSE	1024	

Program code

Modify the following program code to set the security policy in OPC UA:

```
DeviceItem UpcUaSubmodule = ...;
object SecurityPolicies = UpcUaSubmodule.GetAttribute("OpcUaSecurityPolicies");
if (SecurityPolicies | OpcUaSecurityPolicies.OpcUaSecurityPolicies256S ==
OpcUaSecurityPolicies.OpcUaSecurityPolicies256S)
{
//Do something
}
UpcUaSubmodule.SetAttribute("OpcUaSecurityPolicies",
OpcUaSecurityPolicies.OpcUaSecurityPolicies256S |
OpcUaSecurityPolicies.OpcUaSecurityPolicies256SHASE);
```

See also

Opening a project (Page 128)

5.17.3 Accessing OPC UA reference namespace

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access the Reference Namespace and Server Interface (Simatic Interface) subtype. For information on OPC UA SIMATIC Interface see Accessing OPC UA server (SIMATIC) interface (Page 944)

The TIA Portal Openness interface supports the following functionalities of Reference Namespace:

- Create a Reference Namespace
- Import an existing Reference Namespace
- Export an existing Reference Namespace
- Delete an existing Reference Namespace

Method and Property

The Reference Namespace class supports the following method in TIA Portal Openness:

Method	Return type	Definition
Create()	Siemens.Engineering.SW.OpcUa.ReferenceNamespace	To create a new reference namespace object containing the XML provided with the path file
Import()	System.Void	To set the contents of the server interface object based on XML file. The import() parses the XML against the supported schema. If the file can be parsed successfully, the information is added to the reference namespace object, and the import method is successful. If the file cannot be successfully parsed, the contents of the object are unchanged, and the import method will throw an exception.
Export()	System.Void	To write the contents of the reference namespace object to an XML file. If the reference namespace contains no data, an exception will be thrown.
Delete()	System.void	Removes the object from the containing ReferenceNamespaceComposition, and de-allocates the object

The Create() in Reference Namespace class supports the following parameter definition:

Parameter	Return type	Definition
name	System.String	Name for the new reference namespace object
xmlFilePath	System.IO.FileInfo	Path to the XML file for Import

The Import() in Reference Namespace class supports the following parameter definition:

Parameter	Return type	Definition
filePath	System.IO.FileInfo	Wrapper object for the file path of the file to import

The Export() in Reference Namespace class supports the following parameter definition:

Parameter	Return type	Definition
filePath	System.IO.FileInfo	Wrapper object for the file path where the exported file should be written.

The Reference Namespace class provides the following properties:

Property name	Data type	Description	Access
Author	System.string	Author	Read/Write
Comment	Siemens.Engineering.Multilingual.Text	Comment	Read/Write
CreationTime	System.DateTime	Creation time for the reference namespace	Read-only
Enabled	System.Boolean	Enable reference namespace and download to the PLC	Read/Write
LastModified	System.DateTime	Last modified time for the reference namespace	Read-only
Name	System.String	Name	Read-only

Program code

```
// Accessing OPC UA interface;
PlcSoftware software = ....;
OpcUaProvider provider = software.GetService<OpcUaProvider>();
if (provider != null)
{
OpcUaCommunicationGroup commGroup = provider.CommunicationGroup;
ServerInterfaceGroup serverInterfaceGroup = commFolder.ServerInterfaceGroup;
ReferenceNamespaceComposition referenceNamespaces =
serverInterfaceGroup.ReferenceNamespaces;
// Iterating through ReferenceNamespace class

foreach (var referenceNamespace in referenceNamespaces)
{
// ....;
}
}
// Accessing ReferenceNamespace properties
private void accessReferenceNamespaceProperty()
{
RefernceNamespaceComposition referenceNameSpace = ...;
if (referenceNamespaces != null)
{
// Creating reference namespace object containing the xml file provided with name
ReferenceNamespace referenceNamespace = referenceNamespaces.Create(name, xmlFilePath);
referenceNamespace.Author = “XYZ”;
}
}
// Import the reference namespace object
private void importReferenceNamespace()
{
ReferenceNamespaceComposition referenceNamespaces = ...;
if (referenceNamespaces!= null)
{
ReferenceNamespace referenceNamespace = referenceNamespaces.Create(name, xmlFilePath);
if (referenceNamespace != null)
{
referenceNamespace.Import(new FileInfo(importFilePath));
}
}
}
// Export the reference namespace object to XML file
private void exportReferenceNamespace()
{
String exportFilePath = Path.Combine(Directory.GetCurrentDirectory(),
"ExportInterface.xml")
ReferenceNamespace referenceNamespace = …;
referenceNamespace.Export(new FileInfo(exportFilePath));
}
// Remove the reference namespace object
private void deleteReferenceNamespace()
{
ReferenceNamespace referenceNamespace = ...;
referenceNamespace.Delete();
}
```

See also

Opening a project (Page 128)

5.17.4 Accessing OPC UA server (SIMATIC) interface

Requirement

- The application is connected to the TIA Portal using TIA Portal Openness
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access SIMATIC Interface subtype.

The TIA Portal Openness interface supports the following functionalities for SIMATIC Interface:

- Create a SIMATIC Interface
- Export an existing SIMATIC Interface
- Delete an existing SIMATIC Interface

Note

The S7-1200 FW V4.4 will now include support for OPC UA SIMATIC Interfaces.

Method and Property

The SIMATIC Interface class supports the following method in TIA Portal Openness:

Method	Return type	Definition
Create()	Siemens.Engineering.SW.OpcUa.SimaticInterface	To create a OPC UA SIMATIC interface to the PLC
Export()	System.Void	To write the contents of the SIMATIC interface object to an XML file. If the SIMATIC interface contains no data, an exception will be thrown.
Delete()	System.void	Removes the object from the containing Simatic Interface Composition, and de-allocates the object

The Create() in SIMATIC Interface class supports the following parameter definition:

Parameter	Return type	Definition
name	System.String	Name for the new SIMATIC interface object

The Export() in SIMATIC Interface class supports the following parameter definition:

Parameter	Return type	Definition
filePath	System.IO.FileInfo	Wrapper object for the file path where the exported file should be written.

The SIMATIC Interface class supports the following properties in TIA Portal Openness:

Property name	Data type	Description	Access
Author	System.string	Author	Read/Write
Comment	Siemens.Engineering.Multilingual.Text	Comment	Read/Write
CreationTime	System.DateTime	Creation time for the SIMATIC interface	Read-only
Enabled	System.Boolean	Enable SIMATIC interface and download to the PLC	Read/Write
LastModified	System.DateTime	Last modified time for the SIMATIC interface	Read-only
Name	System.String	Name	Read-only

Program code

```
// Accessing OPC UA interface;
PlcSoftware software = ....;
OpcUaProvider provider = software.GetService<OpcUaProvider>();
if (provider != null)
{
OpcUaCommunicationGroup commGroup = provider.CommunicationGroup;
ServerInterfaceGroup serverInterfaceGroup = commFolder.ServerInterfaceGroup;
SimaticInterfaceComposition simaticInterfaces = serverInterfaceGroup.SimaticInterfaces;
// Iterating through Simatic Interfaces class

foreach (var simaticInterface in simaticInterfaces)
{
// ....;
}
}
// Accessing Simatic Interfaces properties
private void accessSimaticInterfaceProperty()
{
SimaticInterfaceComposition simaticInterfaces = ...;
if (simaticInterfaces != null)
{
// create a new empty SIMATIC interface object.
SimaticInterface simaticInterface = simaticInterfaces.Create(name);
simaticInterface.Author = "XYZ";
}
}
// Export the Simatic Interface object to XML file
private void exportReferenceNamespace()
{
String exportFilePath = Path.Combine(Directory.GetCurrentDirectory(),
"ExportInterface.xml")
SimaticInterface simaticInterface = …;
simaticInterface.Export(new FileInfo(exportFilePath));
}
// Remove the Simatic Interface object
private void deleteSimaticInterface()
{
SimaticInterface simaticInterface = …;
simaticInterface.Delete();
}
```

See also

Opening a project (Page 128)

5.17.5 Accessing OPC UA server interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can create server interface objects and import XML files that define the OPC UA server interface for the PLC.

You can use the TIA Portal Openness API to access the following functionality using OpcUaProvider service:

- Adding an OPC UA server interface object
- Enumerating through the server interface belonging to PLC
- Importing a server interface file
- Exporting an existing server interface to XML
- Enabling/ Disabling server interface
- Deleting a server interface

Program code

You can interact with OPC UA server interfaces via an OpcProvider service that can be acquired from PlcSoftware.

```
PlcSoftware software = ....;
OpcUaProvider provider = software.GetService<OpcUaProvider>();
if (provider != null)
{
....
}
```

Modify the following program code to allow navigation to the OPC UA server interfaces:

```
PlcSoftware software = ....;
OpcUaProvider provider = software.GetService<OpcUaProvider>();
if (provider != null)
{
OpcUaCommunicationGroup commGroup = provider.CommunicationGroup;
ServerInterfaceGroup serverInterfaceGroup = commGroup.ServerInterfaceGroup;
ServerInterfaceComposition serverInterfaces = serverInterfaceGroup.ServerInterfaces;
}
```

Modify the following program code to enumerate through the items in the ServiceInterfaceComposition collection:

```
ServerInterfaceComposition serverInterfaces = ....;
foreach (ServerInterface serverInterface in serverInterfaces)
{
....;
}
```

Modify the following program code to add an OPC UA server interface object:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
ServerInterface serverInterface = serverInterfaces.Create(name);
}
```

Modify the following program code to get and set the properties of OPC UA server interface object:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
ServerInterface serverInterface = serverInterfaces.Create("New Interface");
serverInterface.Author = "James Cornett";
}
```

Modify the following program code to set the contents of the server interface object based on XML file:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
ServerInterface serverInterface = serverInterfaces.Create(name);
if (serverInterface != null)
{
serverInterface.Import(new FileInfo(importFilePath));
}
}
```

Modify the following program code to write the contents of the server interface object to an XML file:

```
String exportFilePath = Path.Combine(Directory.GetCurrentDirectory(),
"ExportInterface.xml");
ServerInterface serverInterface = …;
serverInterface.Export(new FileInfo(exportFilePath));
```

Modify the following program code to remove the object from the containing ServerInterfaceComposition, and de-allocates the object:

```
ServerInterface serverInterface = …;
serverInterface.Delete();
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

5.18 SiVArc Openness

5.18.1 Introduction

Introduction

TIA portal openness application allows you to instantiate SiVArc. You must need a client application to access TIA portal, and through openness feature launch SiVArc services. For more details on set up, and accessing Openness, see TIA portal user guide.

Setting up the application

To set up a client application, perform the following steps:

1. Create a console application. Add reference of Public API (`Siemens.Engineering.dll`) available from installed binaries location PublicAPI\V19\Siemens.Engineering.dll
2. Add configuration details to the configuration file. For detailed information of configuration details and steps to access the public API, see TIA openness wiki.
3. To access Sivarc service, use the below mentioned API:

```
using (TiaPortal  tia = new
TiaPortal(TiaPortaMode.WithUserInterface))
{
 Project myProject = tia.Projects.Open(new
FileInfo(@"D:\Project\Project.ap19"));
 Sivarc sivarc =myproject?.GetService<Sivarc>():
 if (sivarc !=null)
  {
  }
}
```

5.18.2 Openness License Handling

When a user opens TIA project through Openness API and makes any SiVArc data changes (Rules and Generation) through Openness API, the system validates the license and if license is not found, '`LicenseNotFound`Exception' recoverable exception is thrown.

5.18.3 SiVArc instance properties

SiVArc instance properties

The table below lists the supported properties and methods for SiVArc:

Property Name	Description	Data Type
AlarmRules	Anchor object for all alarm rule objects	AlarmRulesBrowsable
ScreenRules	Anchor object for all screen rule objects	ScreenRulesBrowsable
TextlistRules	Anchor object for all textlist rule objects	TextlistRulesBrowsable
TagRules	Anchor object for all tag rule objects	TagRulesBrowsable
CopyRules	Anchor object for all copy rule objects	CopyRulesBrowsable
TextlistRules	Enumerate of all immediate first level textlist rules	TextlistRuleComposition
Alarm Rules	Enumerate of all immediate first level alarm rules	AlarmRuleComposition
ScreenRules	Enumerate of all immediate first level screen rules	ScreenRuleComposition
TagRules	Enumerate of all immediate first level tag rules	TagRuleComposition
CopyRules	Enumerate of all immediate first level copy rules	CopyRuleComposition
Groups	Enumerate of all immediate first level alarm rule groups	AlarmRuleGroupComposition
ScreenRu-lesGroups	Enumerate of all immediate first level screen rule groups	ScreenRuleGroupComposition
TextlistGroups	Enumerate of all immediate first level textlist rule groups	TextlistRuleGroupComposition
TagRulesGroups	Enumerate of all immediate first level tag rule groups	TagRuleGroupComposition
CopyRulesGroups	Enumerate of all immediate first level copy rule groups	CopyRuleGroupComposition

Following table lists the AlarmRule and AlarmRuleGroup composition. The same applies to other SiVArc objects.

Method Name	Parameter	Description	Data Type
Find	String- alarm rule /rule-group name	Finds the alarm rule/ alarm rule group from alarm rule/alarm group collection	AlarmRule
CreateFrom	MasterCopy – alarm rule/alarm rule group master copy	Copy alarm rule/alarm rule group master copy from library to project with default replace option	AlarmRule
CreateFrom	MasterCopy – alarm rule/alarm rule group master copy, CreateOptions- Rename/Replace	Copy alarm rule /alarm rule group master copy from library to project with create option	AlarmRule

Openness supports generation and configuration of alarms through alarm rules for Unified devices.

5.18.4 Copying rules or groups from library

Requirement

- Launch TIA portal openness application. For more information on connections, see TIA portal user guide.
- An existing TIA portal project containing Screen rules, screen rule groups, screen rule master copy and screen rule group master copy.

Case 1: When you copy rules/rule groups from master copy to screen rule table

"The `CreateFrom`" API allows you to to copy rule/rule group master copy from library to rule editor. If successful, the API function will return ScreenRule/ScreenRuleGroup. The following code explains how the rules or rule groups are copied from the master copy to the SiVArc rule table:

// Copy rule master copy from library to rule table

```
MasterCopy screenRuleMasterCopy =
project.ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Screen rule_1");
if (screenRuleMasterCopy != null
{
var rule = sivarc.ScreenRules.Tables.Find("Default screen rule
table").Rules.CreateFrom(screenRuleMasterCopy);
if (rule != null)
{
Console.WriteLine("Copied screen rule name - " + rule.Name);
Console.WriteLine("Copied screen rule comment - " + rule.Comment);
Console.WriteLine("Copied screen rule enabled - " + rule.Enabled);
Console.WriteLine("Copied screen rule condition - " + rule.Condition);
}
}
```

By default, the behaviour will be replaced.

Case 2: When you copy rules/rule groups from master copy, the existing rules/rule groups can be renamed or replaced based on the second parameter input

If rules/rule groups in the master copy are already existing in the SiVArc rule table, and if you are trying to copy them, the rules/rule groups gets renamed. The "`CreateOptions`" API will create the rules/rule groups in the SiVArc editor if they are not existing, else they replace the existing rules/rule groups. If successful, the API function will rename ScreenRule/ ScreenRuleGroup. The following code snippet shows the replacing of rules/rule groups:

```
MasterCopy screenRuleMasterCopy =
project.ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Screen rule_1");
if (screenRuleMasterCopy != null)
{
var rule = sivarc.ScreenRules.Tables.Find("Default screen rule
table").Rules.CreateFrom(screenRuleMasterCopy, CreateOptions.Rename);
if (rule != null)
{
Console.WriteLine("Copied screen rule name - " + rule.Name);
Console.WriteLine("Copied screen rule comment - " + rule.Comment);
Console.WriteLine("Copied screen rule enabled - " + rule.Enabled);
Console.WriteLine("Copied screen rule condition - " + rule.Condition);
}
}
```

While creating copy rules, you can choose to browse master copy folders under the Project library and Global library folders. For copying rules, invoke SiVArc using GetService. The `CopyRule` API copies the object into the Master Copy folder using the `MasterCopyFolder`

and `ProjectLibrary` API. The following code snippet shows browsing of folders using copy rule:

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
string projectPath = @"D:\Temp\Project\Project.ap19";
Project project = portal.Projects.Open(new
System.IO.FileInfo(projectPath));
Sivarc sivarc = project.GetService<Sivarc>();
if (sivarc != null)
{
CopyRule copyRule = sivarc.CopyRules.Tables.Find("Default copy rule
table").Rules.Find("Copy rule_1");
MasterCopyFolder masterCopyFolder =
project.ProjectLibrary.MasterCopyFolder.Folders.Find("CopyRuleFolder");
if (masterCopyFolder != null && copyRule != null
{
copyRule.LibraryObject = masterCopyFolder;
}
}
```

For copy rules, you can choose to browse master copy folders under the Project library and Global library folders. For copying rules, invoke SiVArc using GetService. The `CopyRule` API copies the object into the Master Copy folder using the `copyrule.LibraryObject`. System will display exception if null value is set to the `copyrule.LibraryObject` .

5.18.5 Finding a screen rule groups and screen rules

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA portal project consisting of screen rules and screen rule groups.

Finding screen rule groups within screen groups

Case 1:Finding screen rules within screen rule table.

The below API `sivarc.ScreenRules.Rules` allows you to find the available rules within the SiVArc screen rule table as shown below:

```
//Collection of all immediate first level screen rules
ScreenRuleComposition screenRules =
sivarc.ScreenRules.Tables.Find("Default screen rule table").Rules;
if (screenRules != null && screenRules.Count > 0)
{
//Finds screen rule
ScreenRule rule = screenRules.Find("Screen rule_7");
if (rule != null)
{
Console.WriteLine("Screen Rule Name: " + rule.Name);
Console.WriteLine("Screen Rule Comment: " + rule.Comment);
Console.WriteLine("Screen Rule Enabled: " + rule.Enabled);
Console.WriteLine("Screen Rule Condition: " + rule.Condition);
}
}
```

Case 2: Finding screen rule groups within screen rule table.

The below API `sivarc.ScreenRule.Groups` allows you to find the available rules and rule groups within the SiVArc screen rule table as shown below:

```
var groups = sivarc.ScreenRules.Tables.Find("Default screen rule
table").Groups;
if (groups != null && groups.Count > 0)
{
ScreenRule rule = groups.Find("Screen rule_group").Rules.Find("Screen
rule_2");
if (rule != null)
{
Console.WriteLine("Found Screen Rule Name: " + rule.Name);
Console.WriteLine("Found Screen Rule Comment: " + rule.Comment);
Console.WriteLine("Found Screen Rule Enabled: " + rule.Enabled);
Console.WriteLine("Found Screen Rule Condition: " + rule.Condition);
}
var group = groups.Find("Screen rule_group").Groups.Find("Screen rule
group_2");
if(group != null)
{
Console.WriteLine("Found Screen Rule Group Name: " + group.Name);
Console.WriteLine("Found Screen Rule Group Comment: " + group.Comment);
Console.WriteLine("Found Screen Rule Group Enabled: " + group.Enabled);
Console.WriteLine("Found Screen Rule Group Condition: " + group.Condition);
}
}
```

Configuring Condition Operator property

You can configure condition operator property of screen rule within the screen rule table as shown below:

```
ScreenRule screenRule = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Groups.Find("Screen rule
group_1).Rules.Find("Screen  rule_2");

if (screenRule ! = null)

{
```

```
 screenRule.ConditionOperator = ConditionOperator.GreaterThan;
}
```

The `screenRuleGroup.ConditionOperator` API is used to configure the condition operator property within screen rule table.

- For screen rules: when you access screen rule where condition operator property is not applicable, system returns `ConditionOperator.None` value.
  - when you access screen rule where condition operator property is not applicable, system returns `ConditionOperator.None` value
  - When you set some other condition operator value for the above mentioned type of rules, system displays recoverable exception
- For screen rule groups, `ConditionOperator.And`, system will return recoverable exception.

Configuring layout fields

The `GetLayoutFields` API is used to configure layout fields within the screen rules editor as shown below:

```
ScreenRule screenRule = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("Screen rule");
  {
    var layoutFields = screenRule.GetLayoutFields();
    foreach (var layoutField in layoutFields)
     {
        if (layoutField.Equals("TestLayoutField"))
           {
             screenRule.LayoutField = layoutField;
           }
         }
       }
```

The `screenRule.LayoutField` writes the value to layout field.

To update layout fields for screen rule, refer to the below mentioned code:

```
if (screenRule ! = null)
{
  ScreenRule screenRule = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("Screen rule group_1;
 IList<string> layoutFields = screenRule ?. GetLayoutFields() .
ToList();
    {
     if (layoutFields ?.Count > 1)
```

```
        {
     screenRule.LayoutField = layoutfields[0];
    }
}
```

5.18.6 Assigning properties for rules and rule groups

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA project containing screen rules and screen rule groups.

The attributes of rules and rule groups can be set through openness.

Assigning Properties for Screen Rules

To assign properties for screen rules such as Enable, Name, Comment and Condition the following API is used:

```
screenRule.Condition = "StrComp(Block.Name, \"BlockName\")";
```

To assign screen rule properties, it is mandatory that you find the rule within the rule editor. For more information on finding the screen rule, refer section Finding a screen rule groups and screen rules (Page 953)

```
ScreenRule screenRule = sivarc.ScreenRules.Tables.Find("Default screen
rule table").Rules.Find("Screen rule_1");
if (screenRule != null)
{
//Set screen rule attributes
screenRule.Name = "RulenameTest";
screenRule.Comment = "CommentTest";
screenRule.Condition = "StrComp(Block.Name, \"BlockName\")";
screenRule.Enabled = true;
}
```

Assigning Properties for Screen Rule Groups

To assign properties for screen rule groups such as Enable, Name, Comment and Condition the following API is used:

```
Screenrulegroup.Condition = "StrComp(Block.Name, \"BlockName\")";
```

To assign screen rule properties, it is mandatory that you find the rule within the rule editor. For more information on finding the screen rule, refer section Finding a screen rule and screen rule group.

Exceptional cases

SiVArc displays exceptions for the following cases :

- Null/empty value is entered for Name
- Duplicate values entered for Name field
- Value of the Name field exceeds the character limit of 128.
- Name, Comment, Condition (for expected Copy Rules) and Enabled

Below tables lists the APIs used for other Rule types:

Rule Type	APIs
Tag Rule/Tag Rule Group	`TagRule.Condition/TagRuleGroup.Condition`
Text List/ Text List Rule Group	`TextlistRule.Condition/ TextlistRuleGroup.Condition`
Alarm Rule/ Alarm Rule Group	`AlarmRule.Condition/AlarmRuleGroup.Condition`

5.18.7 Creating rule object and rule group

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.

Creating screen rule and screen rule group

To create a screen rule, the `Create` method is used in allignment with `ScreenRuleComposition`. The `Create` API method on `ScreenRule` instance is used for creating rules. The following code explains how the screen rules are created:

```
ScreenRule createdScreenRule = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Create("CreatedScreenRule");
if (createdScreenRule != null)
{
//Created screen rule name property validation
Console.WriteLine("Created screen rule name: " + createdScreenRule.Name);
}
```

If the `CreatedScreenRule` is successful, the `ScreenRule` API will return the newly created screen rule with provided name.

Case1: Create screen rule within a rule group

To create screen rule within an exsiting screen rue group, it is mandatory to find the screen group under which you wish to create a screen rule.

The below API `sivarc.ScreenRules.Groups.Find` allows you to find the screen rule group within the SiVArc screen rule table as shown below:

```
ScreenRule createdScreenRule = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Groups.Find("Screen rule
group_1").Rules.Create("CreatedScreenRuleInUserGroup");
if (createdScreenRule != null)
{
//Created screen rule name property validation
Console.WriteLine("Created screen rule name: " + createdScreenRule.Name);
}
```

For more information on creating screen rule group, refer section Finding screen rules within screen groups.

Exceptions remain same as Assigning Properties for Rules and Rule Groups. For more infornmation on exceptions, refer section Assigning properties for rules and rule groups (Page 956).

Note

Similar methods are used for different rule types. You can replace the object name with the rule type.

5.18.8 Working with screen rule tables and folders

5.18.8.1 Creating screen rule table

Note

The below APIs can also be used in the similar way for alarm rules, copy rules, tag rules, textlist rules.

Case 1: Creating Screen rule table within the Screen rule tables folder

You can create Screen Rule Table within the Screen rule tables folder available under SiVArc project navigation. The `Create` method is used to create a new screen rule table.

```
ScreenRuleTable  screenRuleTable = sivarc.ScreenRules.Tables.Create
(name: "OpnsScreenRuleTable")

If (screenRuleTable ! =null)
{
Console.WriteLine("Created Screen Rule Table : " +
screenRuleTable.Name);
}
```

Case 2: Adding a screen rule table within the user created folder

You can also add a Screen Rule Table within the user created Folder. The `Create` method will be called on ScreenRuleTableComposition.

```
ScreenRuleTable screenRuleTable =
sivarc.ScreenRules.Folders.Find(name: "RuleFolder").Tables.Create
(name :"OpnsScreenRuleTable_1")

If (screenRuleTable ! =null)
{
  Console.WriteLine("Created Screen Rule Table : " +
screenRuleTable.Name);
}
```

5.18.8.2 Creating screen rule folder

Case 1: Creating screen rule folder within the screen rule tables folder

You can create Screen Rule Folder within the Screen rule tables folder available under SiVArc project navigation. The Create method is used to create a new screen rule folder.

```
ScreenRuleFolder screenRuleFolder =
sivarc.ScreenRules.Folders.Create(name:"Folder_ScreenRuleFolder");

If (screenRuleTable ! =null)
{
Console.WriteLine ("Created Screen Rule Folder : " +
screenRuleFolder.Name);
}
```

Case 2: Creating a folder within the user defined folder

You can create a folder within the User defined Folder under Screen rules.

```
ScreenRuleFolder screenRuleFolder = sivarc.ScreenRules.Folders.Find
(name: "RuleFolder").Folders.Create (name:
"Folder_ScreenRuleFolder_1")
{
Console.WriteLine("Created Screen Rule Folder : " +
screenRuleFolder.Name);
}
```

5.18.8.3 Accessing screen rules and groups

To access the screen rules and groups from the RootScreenRuleTable (user created table containing rules and rule groups), use the following code:

```
Console.WriteLine("Rules");

var rules =
sivarc.ScreenRules.Tables.Find("RootScreenRuleTable").Rules;
```

```
if (rules != null && rules.Count > 0)
{
 foreach (var rule in rules)
{
 Console.WriteLine(rule.Name);
}
}

Console.WriteLine("Groups");
var groups =
sivarc.ScreenRules.Tables.Find("RootScreenRuleTable").Groups;
if (groups != null && groups.Count > 0)
{
foreach (var group in groups)
{
Console.WriteLine(group.Name);
}
}
```

To access rules and rule groups from rule table within the folder, use the following code:

```
Console.WriteLine("Rules");
var rules =
sivarc.ScreenRules.Folders.Find("RuleFolder").Tables.Find("AddedScre
enRuleTable").Rules;
if (rules != null && rules.Count > 0)
{
foreach (var rule in rules)
{
Console.WriteLine(rule.Name);
}
}

Console.WriteLine("Groups");
var groups =
sivarc.ScreenRules.Folders.Find("RuleFolder").Tables.Find("AddedScre
enRuleTable").Groups;
if (groups != null && groups.Count > 0)
{
foreach (var group in groups)
{
Console.WriteLine(group.Name);
}
}
```

5.18.8.4 Deleting screen rule table and user defined rule folder

To delete `RootScreenRuleTable` available with Screen rule tables folder, use the following code snippet:

```
sivarc.ScreenRules.Tables.Find(name:
"RootScreenRuleTable").Delete();
```

To delete user defined rule folder, use the following code snippet:

```
sivarc.ScreenRules.Folders.Find(name: "RuleFolder").Delete();
```

Exception:

When user try to delete default screen rule table as mentioned below:

```
sivarc.ScreenRules.Tables.Find(name: "Default screen rule
table").Delete();
```

System will throw recoverable exception with message like default rule table cannot be deleted.

5.18.8.5 Accessing rule table in library type

To access RuleTable in the Library Type folder, the following API is used:

```
LibraryType libraryType =
project.ProjectLibrary.TypeFolder.Types.Find (name:"Screen rule
table")
```

For iterating through different versions in the Library Type, following API is used:

```
LibraryType libraryType =
project.ProjectLibrary.TypeFolder.Types.Find (name:"Screen rule
table")
foreach (var version in LibraryType.Versions)
Console.WriteLine ("Library type {0} version is {1}",
libraryType.Name, version.VersionNumber);
```

5.18.8.6 Creating an instance of rule table

To create an instance of Rule table from Library Type to PNV, CreateFrom method is used. The `CreateFrom` method uses the Rule table composition to instantiate a specific rule table version under Project navigation. `TypeVersion` must be casted to respective rule table. Following code explains the same:

```
LibraryType libraryType =
project.ProjectLibrary.TypeFolder.Types.Find (name:"Screen rule
table");
Version version = new Version ("0.0.2");
var typeVersion = LibraryType.Versions.Find(version);
ruleTableComp.CreateFrom(typeVersion as ScreenRuleTableTypeVersion)
```

Note

Edit operations if attempted using API in release mode, SiVArc rule table throws an exception.

Version details of instantiated Rule Table

To find the version details of rule table which is instantiated under Project navigation, the `LibraryTypeVersion` method is used as shown below:

```
var ruleTableComp = sivarc.ScreenRules.Tables;
ScreenRuleTable screenRuleTable = ruleTableComp.Find(name: "Screen
rule table");
LibraryTypeInstanceInfo instanceInfo =
screenRuleTable.GetService<LibraryTypeInstanceInfo>();
LibraryTypeVersion connectedversion =
instanceInfo.LibraryTypeVersion;
```

5.18.9 Accessing folders in Master copy/Type of a Screen

Accessing folder from Library Type

To access any Library Type folder within the Master copy, `LibraryTypeFolder` API is used as shown below:

```
LibraryTypeFolder libraryTypeFolder =
project.ProjectLibrary.TypeFolder.Folders.Find("Folder");

if (libraryTypeFolder != null && screenRule != null)

{

screenRule.LibraryScreen = libraryTypeFolder;

}
```

Accessing folder from Master copy

The Library Type folder is assigned to Master copy as shown below:

```
MasterCopyFolder masterCopyFolder =
project.ProjectLibrary.MasterCopyFolder.Folders.Find("LegacyItems");

if (masterCopyFolder != null && screenRule != null)

{

    screenRule.LibraryScreen = masterCopyFolder;

}
```

Note

- When you set the value of Master copy/Type of a screen object as folder; and then try setting screen object/layout columns, system will throw an exception.
- After setting the value for the screen object column, if you try browsing for a folder under Master copy/Type column of a screen , SiVArc will delete the already configured values in the screen object and layout field columns.

5.18.10 Accessing library objects

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- Create or use existing screen rules
- TIA project containing master copies of screens, screen objects, program blocks, and faceplate types

Reading and writing program block properties

Case 1: Let us consider reading screen rule's program block attribute. These can be configured by:

- PLC block
- Library type
- Library master copy

The below code explains how to read screen rule's program block attribute from PLC:

```
if (sivarc != null)
{
 ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule")
 ISivarcProgramBlockSource programblock = screenRule?.ProgramBlock;
  if (programBlock ! = null)
   {
     CodeBlock plcBlock = programBlock as CodeBlock;
     Console.WriteLine ("Screen Rule program block: " +
plcBlock?.Name);
 }
}
```

- If you are reading the program block value from library type, ensure to typcast `programBlock` into `LibraryType`
  `LibraryType libraryType = programBlock as LibraryType`
- If you are reading the program block value from library master copy, ensure to typcast `programBlock` into MasterCopy
  `MasterCopy masterCopy = programBlock as MasterCopy`

Case 2: Let us consider writing a screen rule's program block attribute. You can write the value for a program block from

- PLC block
- Library type
- Library master copy

The below code explains how to write screen rule's program block attribute from PLC:

```
if (sivarc != null)
{
ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule")'
Device dev = myProject.Devices.Find("S71500/ET200MP station_1");
softwareContainer softwareContainer = null
var firstorDefault = dev.DeviceItems.FirstorDefault(name =>
name.Name.Equals("PLC_1"));
if (firstorDefault != null)
SoftwareContainer = firstorDefault.GetService<SoftwareContainer>();
if (softwareContainer ! =null)
{
 Software softTarget = softwareContainer.Software;
 PLCSoftware plcSoftware = (PLCSoftware) softTarget;
 var userBlocks = plcSoftware.BlockGroup.Blocks;
 screenRule.ProgramBlock=userBlocks.Find ("Block_2") as CodeBlock;
 }
}
```

- If you are writing the program block value from library type:

```
 screenRule.ProgramBlock =
project.ProjectLibrary.TypeFolder.Types.Find ("Block_Type");
```

- If you are writing the program block value from library master copy:

```
 screenRule.ProgramBlock =
project.ProjectLibrary.MasterCopyFolder.MasterCopies.Find
("Block_2");
```

Note

`ProgramBlock` attribute will be modified for other supported editors too.

Reading and writing screen rule's ScreenObjectLibraryItem property

Case 1: Let us consider reading screen rule's `ScreenObjectLibraryItem` attribute value. You can read the value for a screen object library item from master copy as shown below:

```
if (sivarc != null)
{
 ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule")
 ISivarcLibaryItem screenObjectMasterCopy =
screenRule?.ScreenObjectLibraryItem;
  if (screenObjectMasterCopy ! = null)
   {
     MasterCopy masterCopy= screenObjectMasterCopy as MasterCopy;
     Console.WriteLine ("Screen object master copy name: " +
masterCopy?.Name);
 }
}
```

- If you are reading the screen object from library type, ensure to typecast `screenObjectMasterCopy` into `LibraryType`
  `LibraryTypem libraryType = screenObjectMasterCopy as LibraryType;`

Case 2: Let us consider writing a screen rule's `ScreenObjectLibraryItem` attribute value. You can write the value from library as shown below:

```
if (sivarc != null)
{
 ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule");
 screenRule.ScreenObjectLibraryItem =
project.ProjectLibrary.MasterCopyFolder.MasterCopies.Find
("Button_Lib");
}
```

- If you are writing screen object value from library type:
  `screenRule.ScreenObjectLibraryItem`
  `= project.ProjectLibrary.TypeFolder.Types.Find ("Faceplate_Type");`

Note

Other attributes of Copy rule library object (`LibraryItem`) of type `ISivarcLibraryItem` can be accessed to read/write the value similarly.

Reading and writing screen rules's sceen master property

Case 1: Let us consider reading screen rule's `LibraryScreen` attribute value. You can read the value for a screen library item from master copy as shown below:

```
if (sivarc != null)
{
```

```
 ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule")

 ISivarcLibaryMasterCopy screenLibraryMasterCopy =
screenRule?.LibraryScreen;

  if (screenLibraryMasterCopy ! = null)

   {

     MasterCopy masterCopy= screenLibraryMasterCopy as MasterCopy;

     Console.WriteLine ("Screen master copy name: " +
mastercopy?.Name);

 }

}
```

Case 2: Let us consider writing a screen rule's `LibraryScreen` attribute value. You can write the value for screen rule's screen master copy from library master copy as shown below:

```
if (sivarc != null)

{

 ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule");

 screenRule.LibraryScreen =
project.ProjectLibrary.MasterCopyFolder.MasterCopies.Find
(Screen_lib");

 }
```

Note

Other attributes of type `ISivarcLibraryMasterCopy` (`TextlistLibraryItem` and `AlarmLibraryItem`) can be accessed to read/write the value similarly.

`LibraryScreen` attribute is used to read or write a screen's master copy and type. Using LibraryScreen attribute you can choose to browse for either master copy or type under the **Master copy/ Type of screen** column. The following code snippet shows the navigation under project >project Library > Typefolder> with name "<<screen_type>>". The attribute LibraryScreen is assigned the screen type value.:

```
Sivarc sivarc = project.GetService<Sivarc>();
if (sivarc != null)
{
ScreenRules screenRuleEditor = sivarc.ScreenRules;
if (screenRuleEditor != null)
{
ScreenRuleComposition screenRuleComposition =
                      sivarc.ScreenRules.Tables.Find("Default screen rule
table").Rules;
if (screenRuleComposition != null && screenRuleComposition.Count > 0)
{
ScreenRule scRule = screenRuleComposition.Find("Screen rule");
if (scRule != null)
{
Console.WriteLine((scRule.LibraryScreen as MasterCopy)?.Name);
scRule.LibraryScreen =
project.ProjectLibrary.TypeFolder.Types.Find("Screen_Type");
Console.WriteLine((scRule.LibraryScreen as LibraryType)?.Name);
}
}
}
}
```

5.18.11 Configuring PLC and HMI device

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA portal project consisting of HMI devices, panel devices, and PLCs

Reading and writing PLC/HMI device

Dynamic properties consists of PLC device columns, HMI device columns only.

Case 1: Let us consider reading dynamic properties for rule object. You can read attributes such as plc status, HMI device status as shown in the below code:

To check the status of PLC device if it is selected for a specific rule, you have to pass plc name to `GetAttribute` method.

```
if (sivarc != null)

{

 ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule")

var plcStatus =  screenRule.GetAttribute("PLC_1");

Console.WriteLine ("PLC_1 status: " + plcStatus);

}
```

To check the status of HMI device if it is selected for a specific rule, you have to pass HMI device runtime name to `GetAttribute` method.

For panels, you must pass device name only.

```
var hmiDeviceStatus =  screenRule.GetAttribute("HMI_RT_2");
Console.WriteLine ("HMI_RT_2 status: " + HMIDeviceStatus);
```

To read status for all devices:

```
var devicestatus =  screenRule.GetAttribute new List<string>
 {
   "PLC_1", "PLC_2", "HMI_RT_1", "HMI_RT_2";
 }
foreach (var devicestatus in devicestatus)
{
Console.WriteLine ("PLC & HMI device status: " + devicestatus);
 }
}
```

Case 2: Let us consider writing dynamic properties for rule object. You can read attributes such as PLC device columns to check the status of PLC device if selected for a rule as shown below :

To update the status of PLC device if it is selected for a specific rule, ensure to pass plc name to `SetAttribute` method with boolean value.

```
if (sivarc != null)
{
 ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule")
 screenRule.SetAttribute("PLC_1", false);
}
```

To update the status of HMI device if it is selected for a specific rule, ensure to pass HMI devic runtime name to `SetAttribute` method with boolean value.

For panels, you must pass device name only.

```
if (sivarc != null)
{
 ScreenRule screenRule  = sivarc.ScreenRules.Tables.Find("Default
screen rule table").Rules.Find("ScreenRule");
 screenRule.SetAttribute("HMI_RT_1", true);
}
```

To update status for all devices:

```
if (sivarc != null)
{
 var valuesToBeUpdate = new[]
{
new KeyValuePair<string, object>("PLC_2", true),
new KeyValuePair<string, object>("PLC_3", true),
new KeyValuePair<string, object>("HMI_RT_1", false),
new KeyValuePair<string, object>("HMI_1", true),
};
 ScreenRule screenRule  =
sivarc.ScreenRules.Rules.Find("ScreenRule");
 screenRule.SetAttributes(valuesToBeUpdate);
```

Exceptional cases

- Recoverable exceptional case : While accessing PLC/HMI device columns through `GetAttributes` for other rule object where dynamic attributes are not supported (Tag and Text list rule) , system displays exception Recoverable exception like {'`HMI_RT_1`' is not supported by type '`Siemens.Engineering.SiVArc.TagRule`'}.
- NonRecoverable exception case - For invalid PLC/HMI device names which are not applicable to rule object; system displays `NonRecoverable` exception; valid only for screen, alarm and copy rules.

Note

In case of copy rules, although PLC column is not supported, you can query the PLC status. This occurs due to common behavior of editors, and will not impact the SiVArc functionality.

To Access the PLC /HMI device status for rule/rule group objects, the following dynamic attribute is used: `object status = screenRule.GetAttribute("PLC device name");`

Limitations :

- HMI device names with alphanumeric cannot contain special characters except '_'
- To get HMI device status, prefix the names as "HMI_" followed by the resolved device name. Eg: HMI device name : HMI_@#$RT_1, To access the device name you have to pass as parameter HMI_HMI_RT_1.

- Project consisting of multiple HMI devices with only "_" as special character resolves to same device name, system appends count number along with the HMI device name.Eg: Device_PLC_XYZ representing the first device in project hierarchy.
  Same method is applicable for accessing PLC device statuses.
- Project containing two devices – HMI_@#RT_1 & HMI_%^RT_@#1 ; if both are resolving to same name i.e HMI_RT_1 after removing the special chars (project HMI_@#RT_1 HMI_%^RT_@#). To access the first device status, you have to pass HMI_HMI_RT_1 as parameter and to access the second device status you have to pass HMI_HMI_RT_1_1 as parameter. Same method is applicable for accessing PLC device statuses.

5.18.12 Deleting rules and rule groups

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA project containing screen rules and screen rule groups.

Deleting rules and rule groups

To delete screen rule group, the following API is used:

```
sivarc.ScreenRules.Tables.Find("Default screen rule
table").Rules.Find("Screen rule_1").Delete();
```

The `ScreenRules` is an anchor object for all screen rule objects. To perform deletion, it is mandatory that you find the rule within the rule editor. For more information on finding the screen rule, refer section Finding a screen rule and screen rule group.

To delete screens from screen group, use the following API:

```
sivarc.ScreenRules.Tables.Find("Default screen rule
table").Rules.Find("Screen rule group_1").Delete();
```

5.18.13 Configuring Tag rules

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA portal project consisting of tag rules, and tag rule groups.

Assigning tag rule properties

You can assign tag rule properties such as Name, Condition, Comment, and Enabled. The `TagRule` API is used to set tag rules properties as shown below:

```
TagRule tagrule = sivarc.TagRules.Tables.Find("Default tag rule
table").Rules.Find ("Tag rule_1");
if (tagRule ! = null)
{
  tagRule.Name = "TagRuleName" ;
  tagRule.Comment = "TagRuleComment";
  tagRule.Condition = "StrComp(Block.Name, \"BlockName\")";
  tagRule.Enabled = true;
}
```

Simlarly, `TagRuleGroup` API is used to set the tag rule groups properties.

Note

If the value passed for condition and comment property are more than 500 characters, an exception is displayed.

To update additional properties of tag rules like `TagGroupHierarchy`, `TagTable` and `ConditionalOperator` the following API code is used:

The APIs `TagGroupHierarchy`, `TagTable` and `ConditionOperator` are used to modify the tag groups in hierarchy, tag table, and the condition operator associated for a tag rule in the tag rule editor as shown below:

```
TagRule tagRule = sivarc.TagRules.Tables.Find("Default tag rule
table").Groups.Find("Tag rule group_1").Rules.Find("Tag rule_2");
if (tagRule ! = null)
{
 tagRule.TagGroupHierarchy = "HmiTag.DB.FolderPath";
   tagRule.TagTable = "Hmitag.DB.SymbolicName";
   tagRule.ConditionOperator = ConditonOperator.Equal;
}
```

Exceptional cases

SiVArc displays exceptions for the following cases :

- Null/empty value is entered for `TagTable`
- Duplicate values entered for Name field
- Value of the `TagGroupHierarchy` and `TagTable` field exceeds the character limit of 500
- For invalid values of `ConditionOperator`, `ConditionOperator.None` is returned

See also

Assigning properties for rules and rule groups (Page 956)

5.18.14 Configuring Tag/Text definitions

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- PLC and its associated block

Creating tag definitions

To create tag defintion for a block, you must initially identify the PLC followed by its associated block, and then access SiVArc data provider for that block. Once you acces the SiVArc data provider, use `TagDefinitionComposition` to create a new tag definition. The below code snippet displays the same:

```
CodeBlock block = plcSoftware.BlockGroup.Blocks.Find("Block_1") as
CodeBlock;
// Access SivarcDataProvider service
SivarcDataProvider sivarcDataProvider =
block.GetService<SivarcDataProvider>();
// Create tag definition entry
TagDefinition tagDefinition =
sivarcDataProvider.TagDefinitions.Create("Tag_definition_1");
if (tagDefinition != null)
{
tagDefinition.Value = "Tag_Definition_1_Value";
tagDefinition.Comment = "Tag_Definition_1_Comment";
}
```

Accessing text definitions

To access text definitions for a block, you must initially identify the PLC followed by its associated block, and then access SiVArc data provider for that block. Once you acces the SiVArc data provider, identify for which text definition column you would like the write the values using `TextDefinitionComposition`.

The below code snippet displays the same:

```
TextDefinitionComposition textDefinitions =
sivarcDataProvider.TextDefinitions;
TextDefinition textDefinition = textDefinitions.Find("Text_definition");
if (textDefinition != null)
{
textDefinition.Name = "Updated_Text_definition";
textDefinition.Expression = "Block.DB.SymbolicName";
textDefinition.Comment = "Updated_Text_definition_Comment";
var text = textDefinition.Text;
//Updating german text
Language germanLanguage = project.LanguageSettings.Languages.Find(new
CultureInfo("de-DE"));
text.Items.Find(germanLanguage).Text = "GermanText";
}
```

You can perform similar steps for read operation.

You can also access the configured value for Text definitions which supports multilingual. Choose the language for which you need to read/write values using `"project.LanguageSettings.Languages.Find("<Language name>")"`.

The `MultilingualTextItem` object holds the value for the type of language.

Deleting tag/text definitions

To delete tag definition for a block, you must identify the PLC followed by its associated block, and then access SiVArc data provider for that block. Once you acces the SiVArc data provider, identify the tag definition you would like to delete using `tagDefinition.Delete`.

The same method applies to text definitions, and the `textDefinitions.Delete` API is used for deletion.

The below code snippet displays the same:

```
string projectPath = @"D:\Temp\Project\Project.ap19";
Project project = portal.Projects.Open(new
System.IO.FileInfo(projectPath));
Device myDevice = null;
foreach (var device in project.Devices)
{
//Find device
if (!device.Name.Equals("S71500/200 station_1"))
{
continue;
}
 myDevice = device;
}
if (myDevice?.DeviceItems != null)
{
foreach (var deviceItem in myDevice.DeviceItems)
{
//Find PLC
if (deviceItem.Name == "PLC_1")
{
SoftwareContainer container = deviceItem.GetService<SoftwareContainer>();
Software target = container.Software;
PlcSoftware software = (PlcSoftware) target;
//Find program block
CodeBlock block = software.BlockGroup.Blocks.Find("Block_1") as CodeBlock;
// Access SivarcDataProvider service
SivarcDataProvider sivarcDataProvider =
block.GetService<SivarcDataProvider>();
// Delete tag definition
sivarcDataProvider.TagDefinitions.Find("Tag_definition").Delete();
// Delete text definition
sivarcDataProvider.TextDefinitions.Find("Text_definition").Delete();
}
}
}
```

5.18.15 UMAC set up for openness

About UMAC

The Openness authorization will be used for any modification and no additional UMAC validation is done from SiVArc. For unauthorized users, the system will raise 'FunctionRightNotFoundException' recoverable exception.

5.18.16 SiVArc generation

Requirement

- Launch TIA portal openness application. For more information on connections, see TIA portal user guide.
- An existing TIA portal project connected to an HMI device, and PLC configured.

Important points to note

- Ensure SiVArc license is installed on your PC, else during generation an exception is thrown - "*SiVArc license is missing; it is mandatory to have SiVArc license for modifying data*".
- Ensure valid device name is used, else an exception is thrown - *"HMI device 'deviceName' not found"*.
- Ensure valid PLC name is called, else an exception is thrown - *"PLC device 'plcDeviceName' not found"*.
- Ensure supported device name is called, else an exception is thrown - "*HMI device 'deviceName' is not supported"*
- Ensure supported PLC name is called, else an exception is thrown - "*PLC device 'plcDeviceName' is not supported*"
- Ensure you pass valid GenerationOption parameter. If none is passed, SiVArc generation will happen and by default the TIAP project settings will be used for SiVArc generation
- Ensure you valid PLC name which was not used for generation in the previous generation, else the system freezes.

GenerationOptions- Enum (Flag)

SiVArc supports Enum options, and you can pass combination of two values in the `Generate` API. Following table displays the enum options:

SN	Values	Description
1	None	Nothing is selected, takes default setting for generation
2	AllTags	Generate all tags
3	UsedHmi-Tags	Generate only relevant (used) tags
4	FullGenera-tion	In case when FullGeneration option is not selected, SiVArc will decide internally based on the configuration if it has to perform full generation or delta generation. When you pass `FullGeneration` as parameter, SiVArc will be generated with force full generation.

5	UserCreate-dRules	Only user created rules will be executed
6	EnergySui-teRules	Only EnergySuite rules will be executed
7	AllRules	All rules will be executed

To generate SiVArc, use the following API:

```
sivarc.Generate("HMI_1", new List<string>  {PLC_1},
GenerateOptions.AllTags | GenerateOptions. FullGeneration);
```

SiVArcGenerationResult and SivarcFeedbackMessage

SiVArc generation accesses the following properties on successful generation:

- `IsGenerationSuccessful` - Informs if SiVArc generation is successful.
- `WarningCount` - Total warning count after SiVArc generation
- `ErrorCount` - Total error count after SiVArc generation
- `Messages` - Composition of feedback message

To generate SiVArc result, use the following API:

```
private void WriteSivarcGenerationResults(SivarcGenerationResult result)
{
Console.WriteLine("Is SiVArc generation successful:" +
result.IsGenerationSuccessful);
Console.WriteLine("Tatal warning count:" + result.WarningCount);
Console.WriteLine("Total error count:" + result.ErrorCount);
RecursivelyWriteMessages(result.Messages);
}
```

SiVArc generation accesses the following feedback messages on successful generation:

- Path**:** Header text of feedback message(Header messages will always have empty description field)
- DateTime**:** DateTime of feedback message
- MessageType**:** Feedback message type
- Description**:** Description/Content of Feedback message (Only when Path is empty to ensure not a header message)
- WarningCount**:** Warning count for a header message
- ErrorCount**:** Error count for a header message
- Messages**:** Composition of feedback message (`SivarcFeedbackMessage`)

You can view recursive feedback messages by using the following code snippet:

```
private void RecursivelyWriteMessages(SivarcFeedbackMessageComposition
messages)
{
foreach (SivarcFeedbackMessage message in messages)
{
Console.WriteLine("Path: " + message.Path);
Console.WriteLine("DateTime: " + message.DateTime);
Console.WriteLine("State: " + message.MessageType);
Console.WriteLine("Description: " + message.Description);
Console.WriteLine("Warning Count: " + message.WarningCount);
Console.WriteLine("Error Count: " + message.ErrorCount);
RecursivelyWriteMessages(message.Messages);
}
}
```

SiVArc generation for multiple HMI devices

The `Generate` API allows you to perform SiVArc generation when multiple HMI devices are connected to multiple PLCs with alarm rules, tag rules, text list rules, copy rules, screen rules configured. The `Generate` API method consists of three arguments:

- Devices - Collection of devices for which SiVArc generation is triggered
- PLCs - List of PLCs required for SiVArc generation
- Generate Options - List of settings for SiVArc generation

The below code illustrates the `Generate` method used for generation with mltiple HMI devices and PLCs configured:

```
if (sivarc != null)
{
  SivacGenerationResult result = sivarc.Generate(
  new List<string> {"HMI RT_1", "HMI_RT_2"},
  new List<string> {"PLC_1", "PLC_2"}, GenerateOptions.AllTags |
GenerationOptions.FullGeneration);
  WriteGenerationResults(result);
}
```

- The SivarcGeneration API when executed with "`GenerationOpions.none`" selected, system modifies "All Rules" as default rule set if no rule set is selected means no SiVArc generation performed for selected device and generates HMI objects for all rules.

```
    SivarcGenerationResult result = sivarc.Generate(device
name: "HMI_RT_1", plcs: new List<string>() {"PLC_1"},
GenerationOpions.none);
```

- The SivarcGeneration API when executed with "User created rules" as selection, system considers it as a rule set and generates respective screens with objects.

Note

- If “All Rules” with “User created Rules” or “Energy Suite Rules” is selected, during generation system prioritizes “All rules”, and executes as rule set.
- If “User created Rules” with “Energy Suite Rules” is selected, during generation system prioritizes “User created rules”, and executes as rule set.

Exceptions and special cases

- When `Generate` API is called with null parameter, an exception is thrown - "*The argument 'argument name' may not be null*."
- Ensure to while passing PLCs and HMI devices to the `Generate` API, if any one HMI panel/ device is not connected to PLC, an exception is thrown - "*The HMI/Panel device '{0}' is not connected to the selected PLC devices.”*
- When `Generate` API is called with invalid device name, an exception is thrown - "*HMI device 'deviceName' not found”*.
- When `Generate` API is called with invalid PLC name, an exception is thrown - "*PLC device 'plcDeviceName' not found*."
- When `Generate` API is called with unsupported device name, an exception is thrown - ""HMI device 'deviceName' is not supported."
- When `Generate` API is called with unsupported PLC name, an exception is thrown - "PLC device 'plcDeviceName' is not supported”."
- When `Generate` API is called with list of PLC names , which doesn't contain PLC name that was used in previous SiVArc generation, system will consider the previously frozen and newly passed list of plc device name for generation.
- When `Generate` API is called with list of HMI devices/panels, and each HMI device/panel is connected to a PLC, SiVArc generation will be performed successfully.

5.19 Openness Support

5.19.1 Introduction

Overview

TIA Portal Test Suite via Openness API supports a selection of functions for defined tasks that you can call outside the TIA Portal by means of the TIA Portal Openness API.

You are provided with an overview of the typical programming steps in the sections below. You can learn how the individual code sections interact and how to integrate the respective functions into a complete program. You also get an overview of the code components that have to be adapted for each task.

Connecting to the TIA Portal

TIA Portal Test Suite is an optional package on TIA Portal to increase the efficiency of the PLC programmer by ensuring high quality. To use Test Suite services in the Openness application it is must to install the Test Suite package.

Available namespaces for Test Suite objects

The following namespaces are available for the Test Suite interfaces for features Style guide and Application test:

- `Siemens.Engineering.TestSuite`
- `Siemens.Engineering.TestSuite.StyleGuide`
- `Siemens.Engineering.TestSuite.ApplicationTest`
- `Siemens.Engineering.TestSuite.SystemTest`
- `Siemens.Engineering.TestSuite.StyleGuide.RuleSetExecutor`
- `Siemens.Engineering.TestSuite.ApplicationTest.TestCaseExecutor`
- `Siemens.Engineering.TestSuite.SystemTest.TestCaseExecutor`
- `Siemens.Engineering.TestSuite.TestResults`

Accessing Test Suite as a service

Application

You can access the Test Suite service and its methods in Openness application on the target TIA Portal project using the below code.

Program Code

To access Test Suite as a service:

```
Using Siemens.Engineering;
Using Siemens.Engineering.TestSuite;

TestSuiteService testSuite = project.GetService<TestSuiteService>( ) ;
```

Here 'project' is an instance of the TIA Portal project retrieved in the openness application.

Note

The above program code will give you an build error if the Test Suite package is not installed.

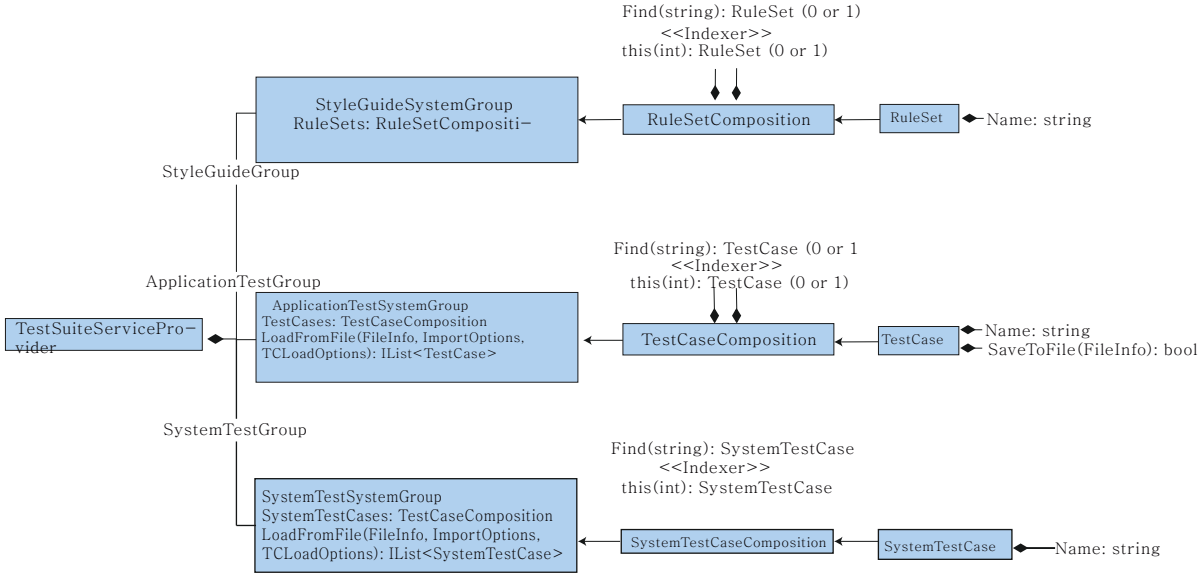
Functions

The section below lists the functions for defined tasks that you can call with TIA Portal Openness outside the TIA Portal.

Test Suite Openness object model

Overview

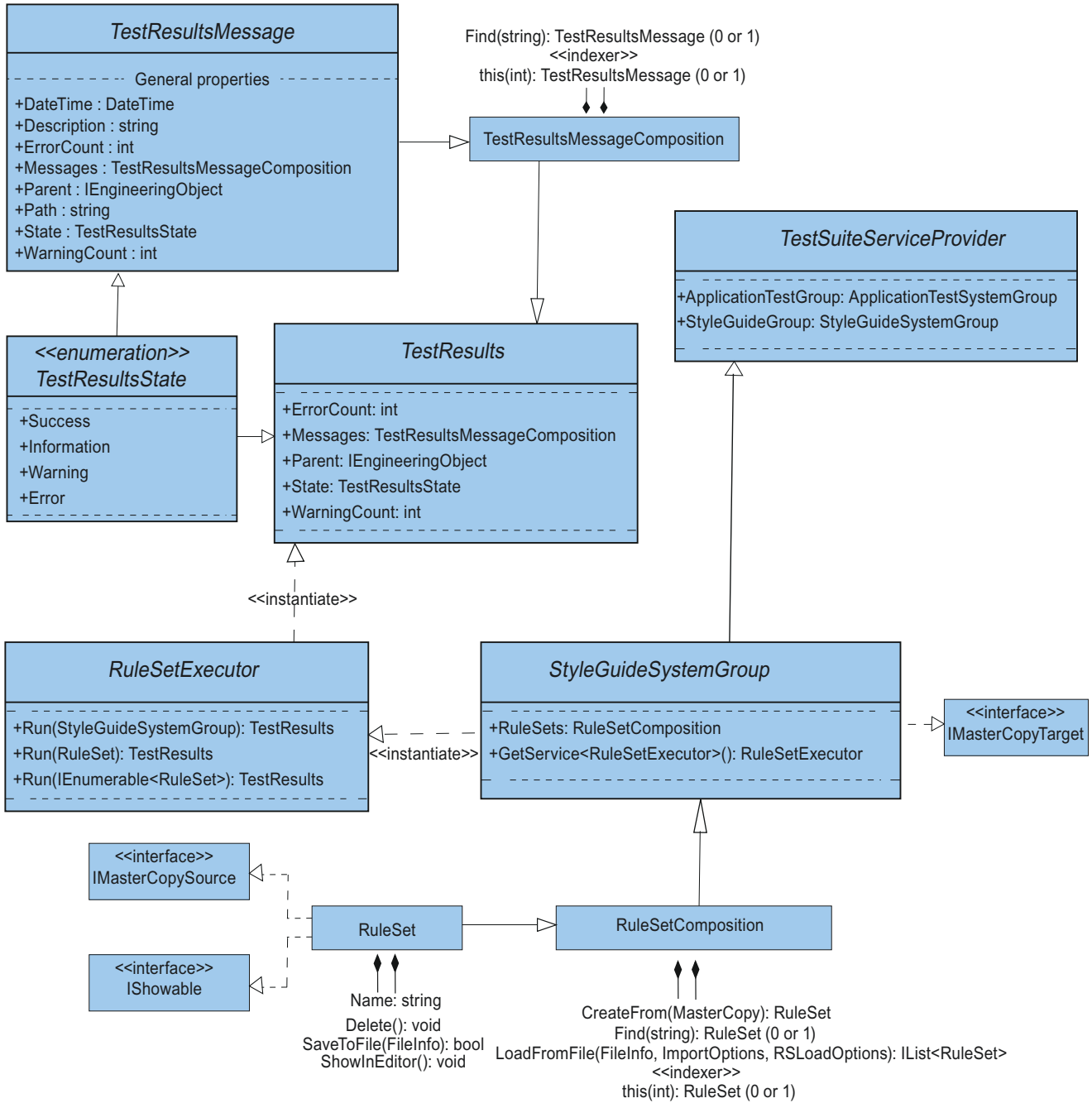
The following diagram describes the highest level of the Test Suite Openness object model:



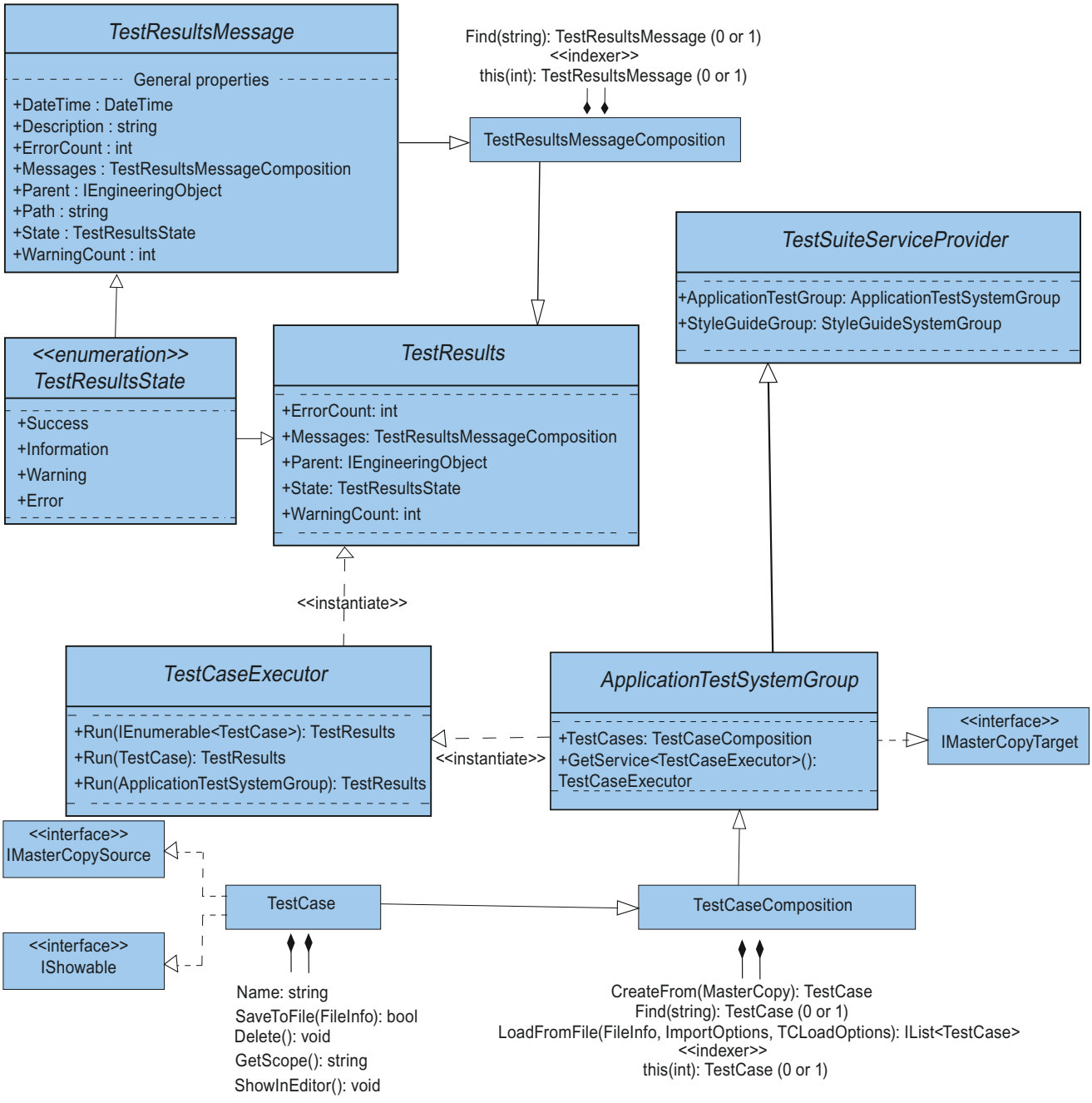
Class diagrams

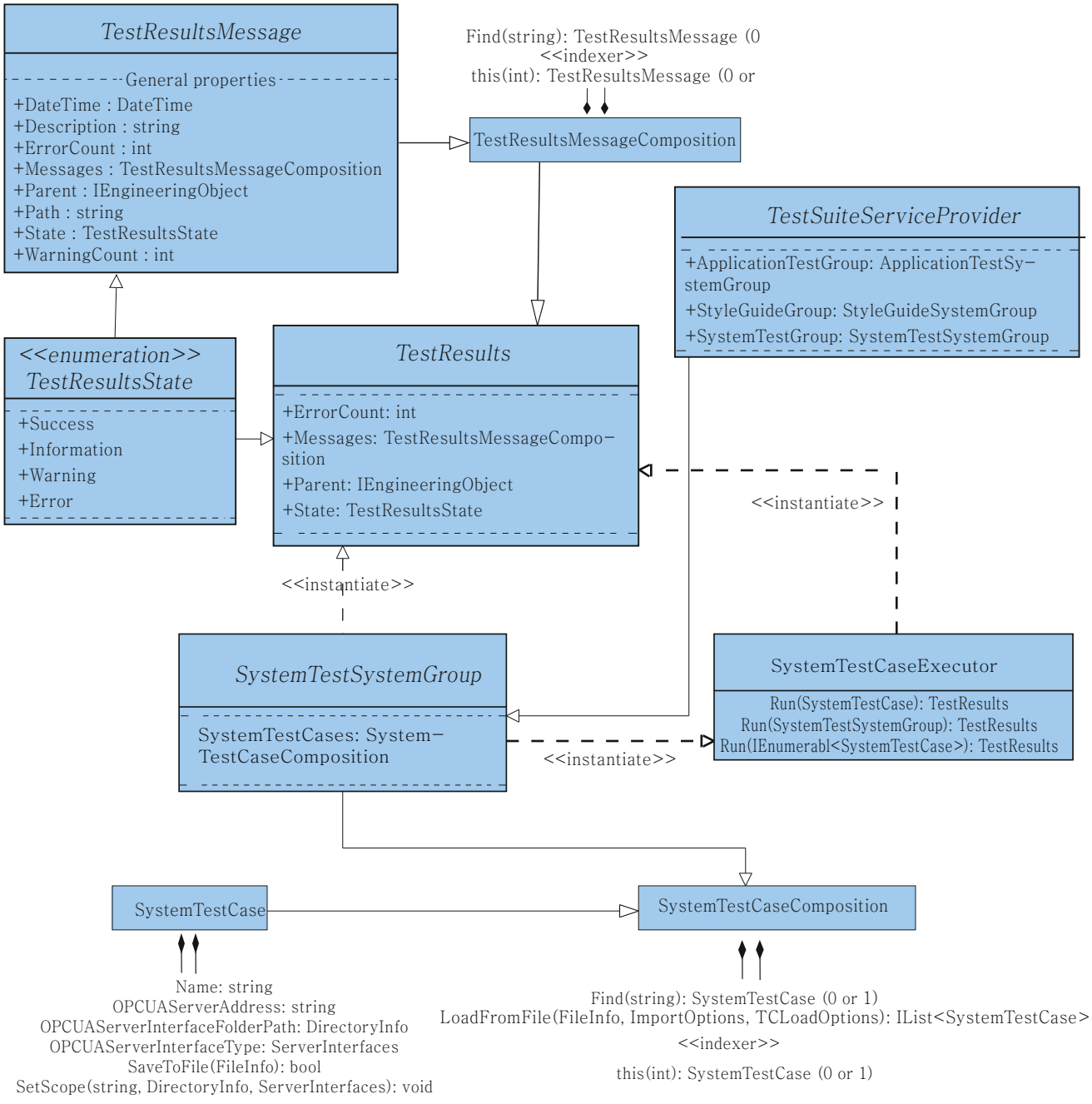
In the TIA Portal Openness object model all classes are defined as abstract which aren't directly instantiated.

The following diagram describes the objects which are located under Rule set execution and master copy.



The following diagram describes the object which are located under Test case execution and master copy.


TestResultsMessage
General properties
+DateTime : DateTime
+Description : string
+ErrorCount : int
+Messages : TestResultsMessageComposition
+Parent : IEngineeringObject
+Path : string
+State : TestResultsState
+WarningCount : int
Find(string): TestResultsMessage (0 or 1)
<<indexer>>
this(int): TestResultsMessage (0 or 1)
TestResultsMessageComposition
TestSuiteServiceProvider
+ApplicationTestGroup: ApplicationTestSystemGroup
+StyleGuideGroup: StyleGuideSystemGroup
<<enumeration>>
TestResultsState
+Success
+Information
+Warning
+Error
TestResults
+ErrorCount: int
+Messages: TestResultsMessageComposition
+Parent: IEngineeringObject
+State: TestResultsState
+WarningCount: int
<<instantiate>>
TestCaseExecutor
+Run(IEnumerable<TestCase>): TestResults
+Run(TestCase): TestResults
+Run(ApplicationTestSystemGroup): TestResults
<<instantiate>>
ApplicationTestSystemGroup
+TestCases: TestCaseComposition
+GetService<TestCaseExecutor>():
TestCaseExecutor
<<interface>>
IMasterCopyTarget
<<interface>>
IMasterCopySource
<<interface>>
IShowable
TestCase
Name: string
SaveToFile(FileInfo): bool
Delete(): void
GetScope(): string
ShowInEditor(): void
TestCaseComposition
CreateFrom(MasterCopy): TestCase
Find(string): TestCase (0 or 1)
LoadFromFile(FileInfo, ImportOptions, TCLoadOptions): IList<TestCase>
<<indexer>>
this(int): TestCase (0 or 1)


TestResultsMessage
General properties
+DateTime : DateTime
+Description : string
+ErrorCount : int
+Messages : TestResultsMessageComposition
+Parent : IEngineeringObject
+Path : string
+State : TestResultsState
+WarningCount : int
Find(string): TestResultsMessage (0
<<indexer>>
this(int): TestResultsMessage (0 or
TestResultsMessageComposition
TestSuiteServiceProvider
+ApplicationTestGroup: ApplicationTestSystemGroup
+StyleGuideGroup: StyleGuideSystemGroup
+SystemTestGroup: SystemTestSystemGroup
<<enumeration>>
TestResultsState
+Success
+Information
+Warning
+Error
TestResults
+ErrorCount: int
+Messages: TestResultsMessageComposition
+Parent: IEngineeringObject
+State: TestResultsState
<<instantiate>>
<<instantiate>>
SystemTestSystemGroup
SystemTestCases: SystemTestCaseComposition
SystemTestCaseExecutor
Run(SystemTestCase): TestResults
Run(SystemTestSystemGroup): TestResults
Run(IEnumerabl<SystemTestCase>): TestResults
<<instantiate>>
SystemTestCase
SystemTestCaseComposition
Name: string
OPCUAServerAddress: string
OPCUAServerInterfaceFolderPath: DirectoryInfo
OPCUAServerInterfaceType: ServerInterfaces
SaveToFile(FileInfo): bool
SetScope(string, DirectoryInfo, ServerInterfaces): void
Find(string): SystemTestCase (0 or 1)
LoadFromFile(FileInfo, ImportOptions, TCLoadOptions): IList<SystemTestCase>
<<indexer>>
this(int): SystemTestCase (0 or 1)

5.19.2 Functions for Style guide

5.19.2.1 Accessing Style guide system folder

Program code

The following properties are used to access the system folder of a Style guide as system group:

```
Using Siemens.Engineering.TestSuite.StyleGuide;

StyleGuideSystemGroup scFolder = testSuite.StyleGuideGroup;
```

Reference

The following table describes the **properties** of the class for Style guide:

Property Name	Data Type	Access Mode
StyleGuideGroup	TestSuite.StyleGuide.StyleGuideSystemGroup	Read

5.19.2.2 Accessing Rule Set composition

Program code

Below property is used to list all available rule sets in the project:

```
RuleSetComposition listRuleSets = testSuite.StyleGuideGroup.RuleSets;
```

Reference

The following table describes the **properties** of the class for Style guide:

Property Name	Data Type	Access Mode
RuleSets	TestSuite.RuleSetComposition	Read

5.19.2.3 Accessing Rule Set

Description

Rule Sets in a TIA Portal project can be retrieved in two ways in openness application as below:

1. Indexed access on RuleSetComposition.
2. Using Find() method on RuleSetComposition.

Program code

Below properties are used to access the single rule set from the project and to retrieve a specific rule set from the project:

```
RuleSet rulesetDB = testSuite.StyleGuideGroup.RuleSets[1];
RuleSet ruleSet1 = testSuite.StyleGuideGroup.RuleSets.Find("rulesetDB");
```

Reference

The following table describes the **properties** of the class for Style guide:

Property Name	Data Type	Access Mode
RuleSet	RuleSetComposition	Read

Methods:

public RuleSet Find(string Name)

5.19.2.4 Accessing Rule Set name

Accessing Rule Set Name

The following property is used to access the rule set name:

```
string ruleSetName = testSuite.StyleGuideGroup.RuleSets[0].Name;
```

Renaming the selected Rule Set with a valid name

Below mentioned property is used to rename the selected rule set with a valid name.

```
public string Name { get; set; }
```

Note

You can perform the following possible tasks with the help of above "Name" property:

- Read the name of the existing rule set
- Change the name of the existing rule set

Reference

The following table describes the property of the class for Style guide:

Property Name	Data Type	Access Mode
Name	String	Read/Write

Program Code

You can modify the following program code to rename a rule set with a valid name:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
RuleSetExecutor styleGuideService =
testSuite.StyleGuideGroup.GetService<RuleSetExecutor>();

//Read the name of a rule set
//Retrieve single rule set
RuleSet myRuleSet1 = testSuite.StyleGuideGroup.RuleSets[0];
String myRuleSet_name = myRuleSet1.Name;

//Modify the name of a rule set
myRuleSet1.Name = "myRuleSet_modified";
```

Note

Application will display an exception if you provide wrong name (For example, starting with whitespace, or containing invalid character).

5.19.2.5 Show Rule Set

Open the selected Rule Set in "WithUI"

Below mentioned property is used to open the selected rule set in TIA Portal editor in the "WithUI" mode.

```
public void ShowInEditor();
```

Program Code

You can modify and use the following program code example to open the rule set in TIA Portal editor in the "WithUI" mode:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing rule set from same project
RuleSet myRuleSet = tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[1];

//Shows the selected item
myRuleSet.ShowInEditor();
```

5.19.2.6 Delete Rule Set

Delete the selected Rule Set in "WithUI/WithoutUI"

Below mentioned property is used to delete the selected rule set from the TIA Portal project in the "withUI/WithoutUI" mode.

```
public void Delete();
```

Program Code

You can modify and use the following program code example to delete the rule set from the TIA Portal project in the "withUI/WithoutUI" mode:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing rule set from same project
RuleSet myRuleSet = tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[1];

//Deletes the selected item
myRuleSet.Delete();
```

5.19.2.7 Modify the Scope of a Rule Set

Updating the Scope for existing Rule Set

Set the Scope of a Rule Set

You can use below API to update the scope of an existing rule set.

Reference

Scope of a rule set can be modified with the below methods:

- `SetScope()`
- `CopyScope()`

The following table describes the syntax of the class:

```
public bool SetScope(UpdateOptions updateOptions,
IList<IEngineeringObject> step7Objects);
```

The following table describes the properties of the class:

Name	Data type	Description
Parameter	`UpdateOptions.Override`	Overwrites the existing scope with the new list of objects available in same project
	`UpdateOptions.Add`	Appends the existing scope with the new list of objects available in same project
	`IList<IEngineeringObject>`	List of objects from the below table can be set as a scope for a rule set

Accessing Step7 objects to assign as a scope to the rule set

Step7 Objects	Representation in opennes application (IEngineeringObject)	Description
Program blocks	`PlcBlockSystemGroup`	All the available blocks under "Program blocks" folder will be set as scope except from "System blocks" folder
PLC tags	`PlcTagTableSystemGroup`	All the available tag tables under "PLC tags" folder will be set as scope
PLC data types	`PlcTypeSystemGroup`	All the available UDTs under "PLC data types" folder will be set as scope except under "System data types"

Software Units	`UnitGroup`
	`UnitGroup.Units[x]`
	`UnitGroup.Units[x].PlcBlockSystemGroup`
	`UnitGroup.Units[x].PlcTagTableSystemGroup`
	`UnitGroup.Units[x].PlcTypeSystemGroup`
		All the available Units under "Software Units" folder will be set as scope which includes Program blocks, PLC tags, and PLC data types. Specific selection is also possible inside Units folder

PLC/CPU	`DeviceItemClassifications.CPU / DeviceItems[0]`	Entire "PLC" with above objects will be set as scope
Project	`Project`	Entire "Project" with above objects will be set as scope
User groups	`PlcBlockSystemGroup.Groups[x]`	Specific user group under "Program blocks" folder can be set as scope
	`PlcTagTableSystemGroup.Groups[x]`	Specific user group under "PLC tags" folder can be set as scope
	`PlcTypeSystemGroup.Groups[x]`	Specific user group under "PLC data types" folder can be set as scope
	`Project.DeviceGroups[0]`	Group of PLCs can be selected as scope under project

Note

Step7 objects which are not supported by Style guide will be ignored during setting the scope and execution of a rule set.

Methods Used

Name of the method	Description	Return Value
`SetScope()`	To modify the scope of an existing rule set	`TRUE`: If the selected supported objects are set as scope
		`FALSE`: If the selected supported objects are not possible to set as scope

Program Code

You can modify and use the following program code example to retrieve the IEngineeringObjects available in the first device item:

```
using Siemens.Engineering;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

// Retrives the first device available in the project
var device = tiaProject.Devices.First();

// To retrieve “PLC_1” object from the above device:
var deviceItem = device.DeviceItems.First(e => e.Classification
==DeviceItemClassifications.CPU && e.Name == "PLC_1");

// To retrieve the software content of the “PLC_1”:
var plc = deviceItem.GetService<SoftwareContainer>().Software as PlcSoftware;

// To retrieve all the blocks available under “Program blocks” folder:
var plc1BlockGroup = plc.BlockGroup;

// To list blocks available under specific user group:
var plc1BlockGroupUserGroups = plc.BlockGroup.Groups[x];

// To provide all the tag tables available under “PLC tags” folder:
var plc1TagTableGroup = plc.TagTableGroup;

// To provide the PLC tag tables available under specific user group in “PLC Tags”
var plc1TagTableGroupUserGroups = plc.TagTableGroup.Groups[x];

// To provide all the UDTs available under "PLC data types" folder:
var plc1DataTypes = plc.TypeGroup;
var plc1DataTypes = plc.TypeGroup.Groups[x];

// To access all "Software units" available in “PLC_1”:
var plc1SoftwareUnit = plc.GetService<PlcUnitProvider>().UnitGroup;

// To provide all the supported objects available in "Entire Project":
ProjectBase Project1 = tiaProject;

// To provide objects from group of devices in a project:
var GroupOfPLC1 = tiaProject.DeviceGroups[x];

// To access rule set from same project
RuleSet ruleSet1 = tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[1];

// update the scope of a ruleSet1:
// The below line of code appends the existing scope with new list of objects:
ruleSet1.SetScope(UpdateOptions.Add, new List<IEngineeringObject> { plc1BlockGroup,
plc1DataTypes, plc1TagTableGroup, plc1SoftwareUnit });

// The below code overwrites the existing scope with new objects:
ruleSet1.SetScope(UpdateOptions.Override, new List<IEngineeringObject>{GroupOfPLC1 });
```

Assigning the Scope from one Rule Set to another Rule Set

You can use below API for copying the Scope from one rule set to another rule set in the same project.

Reference

The following table describes the syntax of the class:

```
public bool CopyScope(RuleSet targRuleSet);
```

The following table describes the properties of the class:

Name	Data type	Description
`targRuleSet`	`RuleSet`	Target rule set to update the scope

Methods used

Name of the method	Description	Return Value
`CopyScope()`	To assign the scope from one rule set to another rule set	`TRUE`: If the scope has been copied successfully
		`FALSE`: If the scope has been failed to copy

Program Code

You can modify and use the following program code example:

```
// To access source and target rule sets from same project:
RuleSet srcRuleSet = tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[1];
RuleSet targRuleSet =
tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[10];

RuleSet srcRuleSet1 =
tiaProject.GetService<TestSuiteService>().StyleGuideGroup.Find("myRS");
RuleSet targRuleSet1 =
tiaProject.GetService<TestSuiteService>().StyleGuideGroup.Find("PLC");

// To assign scope from source to target rule set:
bool retVal = srcRuleSet.CopyScope(targRuleSet);
bool retVal1 = srcRuleSet1.CopyScope(targRuleSet1);
```

5.19.2.8 Execution of Rule Sets

Rule set execution can be performed in three ways in openness application as mentioned below:

1. On specific rule set.
2. On list of rule sets.
3. All available rule sets in Style guide folder.

Reference

Signature

```
public RuleSetExecutor GetService<RuleSetExecutor>();
public TestResults Run(RuleSet ruleSet);
public TestResults Run(new IEnumerable<RuleSet>() {myRuleSet1,myStyleSet2,.. });
public TestResults Run(StyleGuideGroup allRuleSets);
```

The following attributes are supported by TestResults:

Attribute name	Data type	Access mode	Description
Messages	Siemens.Engineering.Test-Suite.TestResultsMessgeCom-position	Read	Composition of feed-back message
Parent	IEngineeringObject	Read	Returns the parent of the container
State	Siemens.Engineering.Test-Suite.TestResultsState	Read	Final state in a test re-sult feedback message with possibly values: Warning, Error and Suc-cess
WarningCount	Int	Read	Number of warnings in a test result feedback message
ErrorCount	Int	Read	Number of errors in a test result feedback message
Description	String	Read	Description or content of a test result feedback message
DateTime	DateTime	Read	Date and Time in a test result feedback mes-sage
Path	String	Read	Path to a test result feedback message

Methods used

Name of the method	Description
`Run()`	To run the selected rule sets

Note

Time Stamp Format

All time stamps are in UTC. If you want to see the local time you can `DateTime.ToLocalTime()`.

Program Code

You can modify and use the following program code example to:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

Project tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
RuleSetExecutor styleGuideService =
testSuite.StyleGuideGroup.GetService<RuleSetExecutor>();

//To execute specific rule set:
//Retrieve single rule set:
//Rule set at 0th index is available in the project:
RuleSet myRuleSet1 = testSuite.StyleGuideGroup.RuleSets[0];

//RuleSet_10 is available in the project:
RuleSet myStyleSet2 = testSuite.StyleGuideGroup.RuleSets.Find("RuleSet_10");

//Execute above retrieved rule sets:
TestResults ruleSetResults1 = styleGuideService.Run(myRuleSet1 );
TestResults ruleSetResults2 = styleGuideService.Run(myStyleSet2 );

//To execute list of rule sets:
//Retrieve list of rule sets to execute:
IEnumerable<RuleSet> ruleSetList = new List<RuleSet>() { myRuleSet1,myStyleSet2 };

//Execute above retrieved list of rule sets:
TestResults ruleSetsResults3 = styleGuideService.Run(ruleSetList );

//To execute all rule sets in StyleGuideGroup:
//Retrieve list of rule sets to execute:
StyleGuideSystemGroup styleGuideFolder = testSuite.StyleGuideGroup;

//Execute all available rule sets in Styleguide folder:
TestResults ruleSetsResults4 = styleGuideService.Run(styleGuideFolder);
```

User can access above test results and iterate through the TestResults messages using the following code:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

...

//Retrieve test results of single rule set execution
WriteTestResults(ruleSetResults1);

//Retrieve test results of list of rule sets execution
WriteTestResults(ruleSetsResults3);

//Retrieve test results of rule sets execution available in
"StyleGuideGroup"
WriteTestResults(ruleSetsResults4);

...

private void WriteTestResults (TestResults result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);

RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(TestResultsMessageComposition
messages,string indent = "")
{
indent += "\t";
foreach (TestResultsMessage message in messages)
{
Console.WriteLine(indent + "Path: " + message.Path);
Console.WriteLine(indent + "DateTime: " + message.DateTime);
Console.WriteLine(indent + "State: " + message.State);
Console.WriteLine(indent + "Description: " + message.Description);
Console.WriteLine(indent + "Warning Count: " + message.WarningCount);
Console.WriteLine(indent + "Error Count: " + message.ErrorCount);

RecursivelyWriteMessages(message.Messages, indent);
}
}
```

5.19.2.9 Import and Export of Rule Sets

Methods used

Following methods are used to import and export the rule sets:

Name of the method	Description
`SaveToFile()`	To export the rule set to an external source file
`LoadFromFile()`	To import the rule set(s) from external source file

Saving a rule set to an external file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `SaveToFile()` method on specific rule set to export it to a selected path.

Signature

```
public void SaveToFile(FileInfo path);
```

The following table shows the needed method parameters:

Name of the parameter	Description
`FileInfo`	Full Path for the rule set to be saved

Program code

Here, `styleChecks` and `styleChecks1` are rule sets available in "MyProject".

```
RuleSetComposition styleChecks = testSuite.StyleGuideGroup.RuleSets;
RuleSetComposition styleChecks1 = testSuite.StyleGuideGroup.RuleSets;
styleChecks.SaveToFile(exportFile);
styleChecks1.SaveToFile(new FileInfo(@"D:\Temp\TSRuleSets1.xml"));
```

Loading rule sets into project from external textual file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `LoadFromFile()` method on the `RuleSetComposition` to import rule set(s) from the selected file.

Signature

```
public IEnumerable<RuleSet> LoadFromFile (FileInfo path,
ImportOptions loadOptions, RSLoadOptions rsLoadOptions);
```

The following table shows the needed method parameters:

Name of the parameter	Description
FileInfo	Full Path for the rule set to be saved
ImportOptions.None	Throws exception if the rule set with the same name exists in the project/if the file contains multiple rule sets with the same name
ImportOptions.Override	Overwrites existing rule set in the project with the same name
RSLoadOptions.IgnorePropertyErrors	Imports the rule set(s) with invalid property values at Author/ Version/Comment
RSLoadOptions.None	Imports the rule set(s) only if there are no warnings/errors in the target file
RSLoadOptions.IgnoreMissingAttributes	Imports the rule set(s) with missing rule attributes by providing the default values to it
RSLoadOptions.SkipInvalidObjects	Imports the rule set(s) by skipping the rules with invalid attributes

Examples:

```
// Throw Exception if the rule set with the same name exists already
// Import the rule set with a invalid properties.

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions.None,
RSLoadOptions.IgnorePropertyErrors);
```

```
// Import rule set with the same name exists already
// Throw exception and abort the import of rule set when there are errors/warnings

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions.None, RSLoadOptions.None);
```

```
// Import rule set with the same name exists already

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions.Override, RSLoadOptions….);
```

```
// Overwrite rule set with the same name exists already in project

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions.Override, RSLoadOptions….);
```

```
// Import rule sets with the default values for the rules with missing attributes

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions…., 
RSLoadOptions.IgnoreMissingAttributes);



// Import rule sets and skip the import for the rules with invalid attributes

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions…., RSLoadOptions.SkipInvalidObjects);



// Import the rule set by replacing missing rule attributes with default values and skipping
rules with invalid attributes

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions…., RSLoadOptions.IgnoreMissingAttributes
| RSLoadOptions.SkipInvalidObjects);



// Imports the rule set(s) with errors in rule set property values, replaces the missing
rule attribute values with the default ones and skips the rules with invalid attributes

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions…., RSLoadOptions.IgnoreMissingAttributes
| RSLoadOptions.SkipInvalidObjects | RSLoadOptions.IgnorePropertyErrors);
```

5.19.2.10 Working with Libraries

Copy Rule Set to Libraries

Rule sets from project can be copied to the libraries for the storage so that it can be reused and modified. And you don't explicitly need to export/import rule sets for external modification.

If the project library is opened in read-write mode, you can create a MasterCopy of an IMasterCopySource at target location.

Reference

The following table describes the syntax of the class:

```
public MasterCopy Create (IMasterCopySource sourceObject);
```

Methods Used

Name of the method	Description
`Create()`	To copy rule set from project to Project/Global libraries

Program Code

You can modify and use the following program code example to copy rule set from project to libraries:

```
using Siemens.Engineering;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;
using Siemens.Engineering.Library.Mastercopies;
...

// Retrieving rule set to copy to libraries:
RuleSet myRuleSet = testSuite.StyleGuideGroup.RuleSets.Find("CalcRuleSet");

// For copying a rule set to project library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Create(myRuleSet );

// For copying a rule set to project library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0]].MasterCopies.Create(myRuleSet );
tiaPortal.GlobalLibraries.Open(new FileInfo((@"D:\Documents\..), OpenMode.ReadWrite);

// For copying a rule set to global library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies.Create(myRuleSet );

// For copying a rule set to global library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies.Create(myRuleSet );
```

Copy Master copies to projects

The TIA Portal Openness API interface supports copying of master copies to project using the `CreateFrom` action. The action will create a new object based on the source master copy and place it in the composition where the action was called. The action will try to create the new rule set/rule sets with the same name as the source master copy. If such name is available in RuleSetComposition, the system will give the new rule set a new name. Then, it will return the new rule set copied.

Reference

The following table describes the syntax of the class:

```
public RuleSet CreateFrom (MasterCopy masterCopy);
```

Methods used

Name of the method	Description
`CreateFrom()`	To copy master copies from Project/Global libraries to project

Program Code

You can modify and use the following program code example to copying master copies from project to libraries:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;
using Siemens.Engineering.Library.Mastercopies;

...

// For accessing master copy object in global library to copy to project:
var masterCopyObjectinGL = tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinGL =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies[1];

// To copy rule set from global library system folder to project:
RuleSet RuleSet1 =testSuite.StyleGuideGroup.RuleSets.CreateFrom(masterCopyObjectinGL);

// To copy rule set from global library user folder to project:
RuleSet RuleSet2
=testSuite.StyleGuideGroup.RuleSets.CreateFrom(masterCopyObjectinUserFolderinGL);

// For accessing master copy object in project library to copy to project:
var masterCopyObjectinPL =tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinPL
=tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0].MasterCopies[0];

// To copy rule set from project library system folder to project:
RuleSet RuleSet3 =testSuite.StyleGuideGroup.RuleSets.CreateFrom(masterCopyObjectinPL);

// To copy rule set from project library user folder to project:
RuleSet RuleSet4
=testSuite.StyleGuideGroup.RuleSets.CreateFrom(masterCopyObjectinUserFolderinPL);
```

Copy Master copies (of type RuleSet) within and between libraries

To copy master copies (of type TestCase) within and between libraries, refer to **TIA Portal Openness API** under chapter **Funtions on libraries** in the topic **Copying master copies** from the Information system.

5.19.3 Functions for Application Test

5.19.3.1 Accessing Application Test system folder

Program code

The following properties are used to access the system folder of a Test Suite as system group:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

TestSuiteService testSuite = project.GetService<TestSuiteService>( ) ;
ApplicationTestSystemGroup atfFolder = testSuite.ApplicationTestGroup;
```

Here 'project' is an instance of the TIA Portal project retrieved in the openness application.

Reference

The following table describes the **properties** of the class for application test:

Property Name	Data Type	Access Mode
ApplicationTestGroup	TestSuite.ApplicationTest.ApplicationTestSystemGroup	Read

5.19.3.2 Accessing Test Case composition

Program code

Below property is used to list all available test cases in the project:

```
TestCaseComposition listTestCases =
testSuite.ApplicationTestGroup.TestCases;
```

Reference

The following table describes the **properties** of the class for application test:

Property Name	Data Type	Access Mode
TestCases	TestSuite.TestCaseComposition	Read

5.19.3.3 Accessing Test Case

Description

Test cases in a TIA Portal project can be retrieved in two ways in openness application as below:

1. Indexed access on TestCaseComposition.
2. Using Find() method on TestCaseComposition.

Program code

Below properties are used to access the single test case from the project and to retrieve a specific test case from the project:

```
TestCase motorFCTestcase = testSuite.ApplicationTestGroup.TestCases[1];
TestCase testCase = testSuite.ApplicationTestGroup.TestCases.Find("TCMot");
```

Reference

The following table describes the **properties** of the class for application test:

Property Name	Data Type	Access Mode
TestCases	TestCaseComposition	Read

Methods:

public TestCase Find(string Name)

5.19.3.4 Accessing Test Case name

Accessing Test Case Name

The following property is used to access the test case name:

```
string testCaseName = testSuite.ApplicationTestGroup.TestCases[0].Name;
```

Renaming the selected Test Case with a valid name

Below mentioned property is used to rename the selected test case with a valid name.

```
public string Name { get; set; }
```

Note

You can perform the following possible tasks with the help of above "Name" property:

- Read the name of the existing test case
- Change the name of the existing test case

Reference

The following table describes the property of the class for application test:

Property Name	Data Type	Access Mode
Name	String	Read/Write

Program Code

You can modify the following program code to rename a test case with a valid name:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

ProjectBase tiaProject = … ;  TestSuiteService testSuite =
tiaProject.GetService<TestSuiteService>();
TestCaseExecutor applicationTestService =
testSuite.ApplicationTestGroup.GetService<TestCaseExecutor>();

//Read the name of a test case
//Retrieve single test case
TestCase myTestCase1 = testSuite.ApplicationTestGroup.TestCases[0];
String myTestCase_name = myTestCase1.Name;

//Modify the name of a test case
myTestCase1.Name = "myTestCase_modified";
```

Note

Application will display an exception if you provide wrong name (For example, starting with whitespace, or containing invalid character).

5.19.3.5 Accessing Test Case Scope

Accessing the Scope of existing Test Case

The following method is used to access the scope of an existing test case:

```
GetScope()
```

The following table describes the syntax of the class:

```
public plcSoftware GetScope();
```

The following table describes the property of the class:

Name	Data type	Description
Parameter	`string`	Name of the PLC, selected as test case scope

Program Code

You can modify and use the following program code example to access the existing test case scope:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing test case from same project
TestCase myTestCase =
tiaProject.GetService<TestSuiteService>().ApplicationTestGroup.TestCases[1];

//Access the scope of myTestCase
PlcSoftware twoHandMonitor = myTestCase.GetScope();
```

5.19.3.6 Show Test Case

Open the selected Test Case in "WithUI"

Below mentioned property is used to open the selected test case in TIA Portal editor in the "WithUI" mode.

```
public void ShowInEditor();
```

Program Code

You can modify and use the following program code example to open the test case in TIA Portal editor in the "WithUI" mode:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing test case from same project
TestCase myTestCase =
tiaProject.GetService<TestSuiteService>().ApplicationTestGroup.TestCases[1];

//Shows the selected item
myTestCase.ShowInEditor();
```

5.19.3.7 Delete Test Case

Delete the selected Test Case in "WithUI/WithoutUI"

Below mentioned property is used to delete the selected test case from the TIA Portal project in the "withUI/WithoutUI" mode.

```
public void Delete();
```

Program Code

You can modify and use the following program code example to delete the test case from the TIA Portal project in the "withUI/WithoutUI" mode:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing test case from same project
TestCase myTestCase =
tiaProject.GetService<TestSuiteService>().ApplicationTestGroup.TestCases[1];

//Deletes the selected item
myTestCase.Delete();
```

5.19.3.8 Modify the Scope of a Test Case

Updating the Scope for existing Test Case

Set the Scope of a Test Case

You can use below API to update the scope of an existing test case.

Reference

Scope of a test case can be modified with the method below:

- `SetScope()`

The following table describes the syntax of the class:

```
public void SetScope(plcSoftware PLC);
public void SetScope(plcSoftware PLC, string instanceName, ExecutionMode tcExecution);
```

The following table describes the properties of the class:

Parameter	Data type	Description
plcSoftware	`plcSoftware`	CPU types are valid objects which are compatible with PLCSIM Advanced
instanceName	`string`	Name of the PLC Advanced Instance
tcExecutionMode	`ExecutionMode`	To select test case execution mode with ExternallyManagedPLCSIMInstance / SystemManagedPLCSIMInstance

Note

It is not possible to set the scope for software controller using SetScope() API

Program Code

You can modify and use the following program code example:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//To retrieve “PLC_1” object from the above device:
var deviceItem = device.DeviceItems.First(e => e.Classification
==DeviceItemClassifications.CPU && e.Name == "PLC_1");

//To retrieve the software content of the “PLC_1”:
var twoHandMonitor = deviceItem.GetService<SoftwareContainer>().Software
as PlcSoftware;

//To access test case from same project
TestCase myTestCase =
tiaProject.GetService<TestSuiteService>().ApplicationTestGroup.TestCases[1
];

//Updates the scope of myTestCase:
myTestCase.SetScope(twoHandMonitor);
```

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Creates a new TIA portal instance
TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface);

//Assigns the path of the project to the project path variable
FileInfo projectPath = new FileInfo(project_path);

//Opens the TIA project AND Test Suite
Project tiaProject = tiaPortal.Projects.Open(projectPath);
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();

//To retrieve “PLC_1” object from the above device:
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");
//To retrieve the software content of the “PLC_1”:
var plc = deviceItem.GetService<SoftwareContainer>().Software as
PlcSoftware;

//Specifies the instance name
string instanceName = "SetScope";

//To access test case from same project
TestCase myTestcase = testSuite.ApplicationTestGroup.TestCases[1];

//Updates the scope of ExternallyManagedPLCSIMInstance:
myTestcase.SetScope(plc, instanceName,
ExecutionMode.ExternallyManagedPLCSIMInstance);
```

5.19.3.9 Execution of Test Cases

Test case execution can be performed in three ways in openness application as mentioned below:

1. On specific test case.
2. On list of test cases.
3. All available test cases in Application test folder.

Reference

Signature

```
public TestCaseExecutor GetService<TestCaseExecutor>();
public TestResults Run(TestCase myTestCase);
public TestResults Run(new IEnumerable<TestCase>() {myTestCase1,myTestCase2,.. });
public TestResults Run(ApplicationTestGroup allTestCases);
```

The following attributes are supported by TestResults:

Attribute name	Data type	Access mode	Description
Messages	Siemens.Engineering.TestSuite.TestResultsMessgeComposition	Read	Composition of feedback message
Parent	IEngineeringObject	Read	Returns the parent of the container
State	Siemens.Engineering.TestSuite.TestResultsState	Read	Final state in a test result feedback message with possibly values: Warning, Error and Success
WarningCount	Int	Read	Number of warnings in a test result feedback message
ErrorCount	Int	Read	Number of errors in a test result feedback message
Description	String	Read	Description or content of a test result feedback message
DateTime	DateTime	Read	Date and Time in a test result feedback message
Path	String	Read	Path to a test result feedback message

Methods used

Name of the method	Description
`Run()`	To run the selected test case(s)

Note

Time Stamp Format

All time stamps are in UTC. If you want to see the local time you can `DateTime.ToLocalTime()`.

Note

You can run the test cases in both ExternallyManagedPLCSIMInstance or SystemManagedPLCSIMInstance configured mode of test case execution.

For information on ExternallyManagedPLCSIMInstance refer Test Case Execution using ExternallyManagedPLCSIMInstance

Program Code

You can modify and use the following program code example:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
TestCaseExecutor applicationTestService =
testSuite.ApplicationTestGroup.GetService<TestCaseExecutor>();

//To execute specific test case:
//Retrieve single test case:
//Test case at 0th index is available in the project:
TestCase myTestCase1 = testSuite.ApplicationTestGroup.TestCases[0];

//TestCase_10 is available in the project:
TestCase myTestCase2 =
testSuite.ApplicationTestGroup.TestCases.Find("TestCase_10");

//Execute above retrieved test cases:
TestResults TestCaseResults1 = applicationTestService.Run(myTestCase1 );
TestResults TestCaseResults2 = applicationTestService.Run(myTestCase2 );

//To execute list of test cases:
//Retrieve list of test cases to execute:
IEnumerable<TestCase> TestCaseList = new List<TestCase>()
{ myTestCase1,myTestCase2 };

//Execute above retrieved list of test cases:
TestResults TestCasesResults3 = applicationTestService.Run(TestCaseList );

//To execute all test cases in ApplicationTestGroup:
//Retrieve list of test cases to execute:
ApplicationTestSystemGroup applicationTestFolder =
testSuite.ApplicationTestGroup;

//Execute all available test cases in ApplicationTestFolder:
TestResults TestCasesResults4 =
applicationTestService.Run(applicationTestFolder);
```

User can access above test results and iterate through the TestResults messages using the following code:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

...

//Retrieve test results of single test case execution
WriteTestResults(TestCaseResults1);

//Retrieve test results of list of test cases execution
WriteTestResults(TestCasesResults3);

//Retrieve test results of test cases execution available in "ApplicationTestGroup"
WriteTestResults(TestCasesResults4);

...

private void WriteTestResults (TestResults result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);

RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(TestResultsMessageComposition messages,string indent
= "")
{
indent += "\t";
foreach (TestResultsMessage message in messages)
{
Console.WriteLine(indent + "Path: " + message.Path);
Console.WriteLine(indent + "DateTime: " + message.DateTime);
Console.WriteLine(indent + "State: " + message.State);
Console.WriteLine(indent + "Description: " + message.Description);
Console.WriteLine(indent + "Warning Count: " + message.WarningCount);
Console.WriteLine(indent + "Error Count: " + message.ErrorCount);

RecursivelyWriteMessages(message.Messages, indent);
}
}
```

You can copy and modify the below sample code to run test cases in "ExternallyManagedPLCSIMInstance mode".

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;
using Siemens.Simatic.Simulation.Runtime;

string m_PlcInstanceName;
IInstance m_PlcSimInstance;



//To create PLCSIM Advanced instance and power on

m_PlcSimInstance = SimulationRuntimeManager.RegisterInstance(ECPUType.CPU1500_Unspecified,
m_PlcInstanceName);
m_PlcSimInstance.PowerOn(60000);

ProjectBase tiaProject = … ;

//Download the selected device item (Project PLC to PLCSIM instance created)

//Refer the chapter TIA Portal Openness API->Functions for accessing the data of a PLC
device->Functions for downloading data to PLC device ->Downloading to PLC devices

DeviceItem deviceItem = ...;
DownloadProvider downloadProvider = deviceItem.GetService<DownloadProvider>();
      if (downloadProvider != null)
                {
                      ...

                 }

//To retrieve the software content of the selected PLC to set the scope of test cases to run

var twoHandMonitor = deviceItem.GetService<SoftwareContainer>().Software as PlcSoftware;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
TestCaseExecutor applicationTestService =
testSuite.ApplicationTestGroup.GetService<TestCaseExecutor>();

//To run list of test cases:

//Test case at 0th index is available in the project:

TestCase myTestCase1 = testSuite.ApplicationTestGroup.TestCases[0];
myTestCase1.SetScope(twoHandMonitor,m_PlcInstanceName,ExecutionMode.ExternallyManagedPLCSI
MInstance);

//TestCase_10 is available in the project:

TestCase myTestCase2 = testSuite.ApplicationTestGroup.TestCases.Find("TestCase_10");
myTestCase2.SetScope(twoHandMonitor,m_PlcInstanceName,ExecutionMode.ExternallyManagedPLCSI
MInstance);

//Retrieve list of test cases to execute:

IEnumerable<TestCase> TestCaseList = new List<TestCase>() { myTestCase1,myTestCase2 };
```

```
//Execute above retrieved list of test cases, Connects to the exisiting PLCSIM Advanced
instance and runs the test cases selected

TestResults TestCasesResults = applicationTestService.Run(TestCaseList );
```

Working with PLCSIM Advanced V6.0

Working with TIA Portal in legacy mode

You can modify and use the following program code example:

```
//Initialize TIA Portal instance
var tiaPortalUi = new TiaPortal(TiaPortalMode.WithUserInterface);

//Open the project
var project = tiaPortalUi.Projects.Open(new FileInfo(projectPath));

//Enable legacy mode for the PLC

//For R/H type handling
var onlineProvider = device.GetService<RHOnlineProvider>();
var configuration = onlineProvider.Configuration;
if(!configuration.EnableLegacyCommunication) configuration.EnableLegacyCommunication =
true;

//For any other CPU type
var onlineProvider = deviceItem.GetService<OnlineProvider>();
var configuration = onlineProvider.Configuration;
if(!configuration.EnableLegacyCommunication) configuration.EnableLegacyCommunication =
true;

//Execute test case
var testSuiteService = project.GetService<TestSuiteService>();
var testCaseExecutor =
testSuiteService.ApplicationTestGroup.GetService<TestCaseExecutor>();
testCaseExecutor.Run(...);
```

Working with TIA Portal in secure mode

You can modify and use the following program code example:

```
//Initialize TIA Portal instance
var tiaPortalUi = new TiaPortal(TiaPortalMode.WithUserInterface);

//Open the project
var project = tiaPortalUi.Projects.Open(new FileInfo(projectPath));

//Enable secure mode for the PLC

//For R/H type handling
var onlineProvider = device.GetService<RHOnlineProvider>();
onlineProvider.Configuration.OnlineLegitimation += Configuration_OnlineLegitimation;

private void Configuration_OnlineLegitimation(OnlineConfiguration onlineConfiguration)
{
var tlsCommunication = onlineConfiguration as TlsVerificationConfiguration;
if(tlsCommunication == null) return;
tlsCommunication.CurrentSelection = TlsVerificationConfigurationSelection.Trusted;
}

//For any other CPU type
var onlineProvider = project.Devices[0].DeviceItems[1].GetService<OnlineProvider>();
onlineProvider.Configuration.OnlineLegitimation += Configuration_OnlineLegitimation;

private void Configuration_OnlineLegitimation(OnlineConfiguration onlineConfiguration)
{
var tlsCommunication = onlineConfiguration as TlsVerificationConfiguration;
if(tlsCommunication == null) return;
tlsCommunication.CurrentSelection = TlsVerificationConfigurationSelection.Trusted;
}
```

5.19.3.10 Import/Export of test cases using openness APIs

Methods used

Following methods are used to import and export the test cases:

Name of the method	Description
`SaveToFile()`	To export the test case
`LoadFromFile()`	To import the test case(s)

Saving a test case to an external textual file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `SaveToFile()` method on specific test case to export it to a selected path.

Signature

```
public void SaveToFile(FileInfo path);
```

The following table shows the needed method parameters:

Name of the parameter	Description
`FileInfo`	Full Path for the test case to be saved

Program code

Here, `motorFBTestCase` and `motorFBTestCase1` are test cases available in "MyProject".

```
motorFBTestCase.SaveToFile(exportFile);
motorFBTestCase1.SaveToFile(new FileInfo(@"D:\Temp\TSTestcases2.tat"));
```

Loading test cases into project from external textual file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `LoadFromFile()` method on the `TestCaseComposition` to import a test case(s) from the selected file.

Signature

```
public IEnumerable<TestCase> LoadFromFile (FileInfo path,
ImportOptions loadOptions, TCLoadOptions tcLoadOptions);
```

The following table shows the needed method parameters:

Name of the parameter	Description
`FileInfo`	Full Path for the test case to be saved
`ImportOptions.None`	Throws exception if the test case with the same name exists in the project/if the file contains a test case with the same name more than once
`ImportOptions.Override`	Overwrites existing test case in the project with the same name
`tcLoadOptions.IgnoreInvalidObject`	Import the test case if the scope/property in the file is not valid
`tcLoadOptions.None`	Throws exception if the test case imports with invalid scope/ invalid properties

Program code

Here, `motorFBTestCases` is an instance of a `TestCaseComposition`.

Examples:

```
// Load test cases from above obtained file path to project
// Throw Exception if the test case with the same name exists already
// Import the test case with a invalid scope/properties

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tat");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.None,
TCLoadOptions.IgnoreInvalidObject);
```

```
// Load test cases from above obtained file path to project
// Throw Exception if the test case with the same name exists already
// Throw Exception if the test case created with a invalid scope/invalid properties

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tat");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.None, TCLoadOptions.None);
```

```
// Load test cases from above obtained file path to project
// Import test case with the same name exists already
// Import the test case with a invalid scope/properties

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tat");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.Override,
TCLoadOptions.IgnoreInvalidObject);
```

```
// Load test cases from above obtained file path to project
// Overwrite test case with the same name exists already in project
// Throw Exception if the test case created with a invalid scope /invalid properties

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tat");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.Override, TCLoadOptions.None);
```

5.19.3.11 Working with Libraries

Copy Test Case to Libraries

Test cases from project can be copied to the libraries for the storage so that it can be reused and modified. And you don't explicitly need to export/import test cases for external modification.

If the project library is opened in read-write mode, you can create a MasterCopy of an IMasterCopySource at target location.

Reference

The following table describes the syntax of the class:

```
public MasterCopy Create (IMasterCopySource sourceObject);
```

Methods Used

Name of the method	Description
`Create()`	To copy test case from project to Project/Global libraries

Program Code

You can modify and use the following program code example to copy test case from project to libraries:

```
using Siemens.Engineering;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;
...

// Retrieving test case to copy to libraries:
TestCase myTestCase = testSuite.ApplicationTestGroup.TestCases.Find("CalcTestCase");

// For copying a test case to project library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Create(myTestCase );

// For copying a test case to project library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.Folders[x].MasterCopies.Create(myTestCase );
tiaPortal.GlobalLibraries.Open(new FileInfo((@"D:\Documents\.."), OpenMode.ReadWrite);

// For copying a test case to global library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies.Create(myTestCase );

// For copying a test case to global library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[x].MasterCopies.Create(myTestCase );
```

Copy Master copies to projects

The TIA Portal Openness API interface supports copying of master copies to project using the `CreateFrom` action. The action will create a new object based on the source master copy and place it in the composition where the action was called. The action will try to create the new test case/test cases with the same name as the source master copy. If such name is available in TestCaseComposition, the system will give the new test case a new name. Then, it will return the new test case copied.

Reference

The following table describes the syntax of the class:

```
public TestCase CreateFrom (MasterCopy masterCopy);
```

Methods used

Name of the method	Description
`CreateFrom()`	To copy master copies from Project/Global libraries to project

Program Code

You can modify and use the following program code example to copying master copies from project to libraries:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

...

// For accessing master copy object in global library to copy to project:
var masterCopyObjectinGL = tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinGL =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies[1];

// To copy test case from global library system folder to project:
TestCase testCase1
=testSuite.ApplicationTestGroup.TestCases.CreateFrom(masterCopyObjectinGL);

// To copy test case from global library user folder to project:
TestCase testCase2
=testSuite.ApplicationTestGroup.TestCases.CreateFrom(masterCopyObjectinUserFolderinGL);

// For accessing master copy object in project library to copy to project:
var masterCopyObjectinPL =tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinPL
=tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0].MasterCopies[0];

// To copy test case from project library system folder to project:
TestCase testCase3
=testSuite.ApplicationTestGroup.TestCases.CreateFrom(masterCopyObjectinPL);

// To copy test case from project library user folder to project:
TestCase testCase4
=testSuite.ApplicationTestGroup.TestCases.CreateFrom(masterCopyObjectinUserFolderinPL);
```

Copy Master copies (of type TestCase) within and between libraries

To copy master copies (of type TestCase) within and between libraries, Refer to **TIA Portal Openness API** under chapter **Funtions on libraries** in the topic **Copying master copies** from the Information system.

5.19.4 Functions for System Test

5.19.4.1 Accessing the System Test system folder

Program code

The following properties are used to access the system folder of a System test as system group:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
TestSuiteService testSuite = project.GetService<TestSuiteService>( );
SystemTestSystemGroup stFolder = testSuite.SystemTestGroup;
```

Here 'project' is an instance of the TIA Portal project retrieved in the openness application.

Reference

The following table describes the **properties** of the class for system test:

Property Name	Data Type	Access Mode
SystemTestGroup	TestSuite.SystemTest.SystemTestSystemGroup	Read

5.19.4.2 Accessing Test Case Composition

Program code

Below property is used to list all available system test cases in the project:

```
SystemTestCaseComposition listTestCases = testSuite.SystemTestGroup.SystemTestCases;
```

Reference

The following table describes the **properties** of the class for system test:

Property Name	Data Type	Access Mode
SystemTestCases	TestSuite.SystemTestCaseComposition	Read/Write

5.19.4.3 Accessing System Test Case

Description

Test cases in a TIA Portal project can be retrieved in two ways in openness application as below:

1. Indexed access on SystemTestCaseComposition.
2. Using Find() method on SystemTestCaseComposition.

Program code

Below properties are used to access the single test case from the project and to retrieve a specific test case from the project:

```
SystemTestCase motorFCTestcase = testSuite.SystemTestGroup.SystemTestCases[1];
SystemTestCase testCase = testSuite.SystemTestGroup.SystemTestCases.Find("TCMot");
```

Reference

The following table describes the properties of the class for system test:

Property Name	Data Type	Access Mode
SystemTestCases	TestSuite.SystemTestCaseComposition	Read/Write

Methods:

```
public SystemTestCase Find(string Name)
```

5.19.4.4 Accessing System Test Case name

Accessing System Test Case Name

The following property is used to access the test case name:

```
string testCaseName = testSuite.SystemTestGroup.SystemTestCases[0].Name;
```

Renaming the selected Test Case with a valid name

Below mentioned property is used to rename the selected test case with a valid name.

```
public string Name { get; set; }
```

Note

You can perform the following possible tasks with the help of above "Name" property:
- Read the name of the existing test case
- Change the name of the existing test case

Reference

The following table describes the property of the class for system test:

Property Name	Data Type	Access Mode
Name	String	Read/Write

Program Code

You can modify the following program code to rename a test case with a valid name:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
SystemTestCaseExecutor systemTestService =
testSuite.SystemTestGroup.GetService<SystemTestCaseExecutor>();
//Retrieve single test case
SystemTestCase myTestCase1 = testSuite.SystemTestGroup.SystemTestCases[0];
//Read the name of a test case
String myTestCase_name = myTestCase1.Name;
```

5.19.4.5 Accessing System Test Case Scope

Accessing the Scope of existing Test Case

The selected test case scope can be retrieved using the below mentioned test case properties

Syntax is as follows:

```
public string OPCUAServerAddress { get;}
public DirectoryInfo OPCUAServerInterfaceFolderPath{ get;}
public ServerInterfaces OPCUAServerInterfaceType { get;}
```

The following table describes the parameters:

Name	Data Type	Description
OPCUAServerAddress	String	OPC UA server address configured to the test case
OPCUAServerInterfaceFolderPath	DirectoryInfo	Directory information for the server interface files mapped
OPCUAServerInterfaceType	Siemens.Engineering.TestSuite.SystemTest.ServerInterfaces	Server interface type (UserDefined / standardSIMATIC / SiOMECompanionSpecification) configured.

Program Code

You can modify and use the following program code example to access the existing system test case scope:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
SystemTestCase tcForModule1 = testSuite.SystemTestGroup.SystemTestCases[0];
//Retrieve the OPC UA server address configured for a test
casestring tcForModule1OPCUAServeraddress = tcForModule1.OPCUAServerAddress;
//Retrieve the directory information of the OPC UA server interfaces mapped for a test
caseDirectoryInfo tcForModule1ServerInterfaces =
tcForModule1.OPCUAServerInterfaceFolderPath;
//Retrieve the full folder path of the OPC UA server interfaces mapped for a test
casestring  tcForModule1ServerInterfacesPath = tcForModule1ServerInterfaces.FullName;
//Validate the folder path of the OPC UA server interfaces mapped for a test
caseDirectoryInfo directoryInfo=new DirectoryInfo(@"D:\TestSuite\SystemTest\PACKMLFiles");
Assert.AreEqual(systemTestCase.OPCUAServerInterfaceFolderPath, directoryInfo);
// Retrieve the OPC UA server interface selected for a test case
ServerInterfcaes tcForModule1ServerInterfaceType = tcForModule1.OPCUAServerInterfaceType;
```

5.19.4.6 Modify the Scope of a System Test Case

Set the Scope of a System Test Case

You can use below API to modify the scope of an existing system test case.

Syntax of the methods is as follows:

```
public void SetScope ( string OPCUAServerAddress, ServerInterfaces
OPCUAServerInterfaceType, DirectoryInfo OPCUAServerInterfaceFolderPath );
```

Here the scope can be set using the OPCUAServerAddress, OPCUA ServerInterfaceType and OPCUA Server Interface Folder Path.

As the folder path is not mandatory run the test case you can use the following syntax modify the scope without parameter *"OPCUAServerInterfaceFolderPath"*

```
public void SetScope ( string OPCUAServerAddress, ServerInterfaces
OPCUAServerInterfaceType);
```

The following table describes the parameters:

Name	Data Type	Description
OPCUAServerAddress	String	OPC UA server address configured to the test case
OPCUAServerInterfaceFolderPath	DirectoryInfo	Directory information for the server interface files mapped
OPCUAServerInterfaceType	Siemens.Engineering.TestSuite.SystemTest.ServerInterfaces	Server interface type (UserDefined / standardSIMATIC /SiOMECompanionSpecification) configured.

Note

You can also assign the scope of one test case to another test case. Refer to the following program code for further information.

Note

Test case will be automatically refreshed/updated when you modify the scope of the test case. For futher information refer System test case scope update

Program Code

You can modify and use the following program code example:

```
using Siemens.Engineering;
……..
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;

// Accessing System test case
SystemTestCase tcModule1 =
tiaProject.GetService<TestSuiteService>().SystemTestGroup.SystemTestCases[1];

//  OPC UA server address
string targetPLCasOPCUAServer = “opc.tcp://opcua.demo-setscope.com:51210/UA/SampleServer”;

//Folder path to OPC UA server interfaces stored
DirectoryInfo tcModule1ServerInterafcesPath = new
DirectoryInfo(@"D:\Machines\ModuleTest\ServreInterafces");

//Setting the scope of a selected test case
tcModule1.SetScope(targetPLCasOPCUAServer ,ServerInterfaces.StandardSIMATIC ,
tcModule1ServerInterafcesPath);

**********************************************************************
Assigning the scope of one test case to another test case
**********************************************************************

SystemTestCase tcForModule3 = testSuite.SystemTestGroup.SystemTestCases[3];
SystemTestCase tcForModule4 = testSuite.SystemTestGroup.SystemTestCases[4];

//Retrieve the OPC UA server address configured for a “tcForModule3”

string tcForModule3OPCUAServeraddress = tcForModule3.OPCUAServerAddress;

//Retrieve the directory information of the OPC UA server interfaces mapped for a
“tcForModule3”

DirectoryInfo tcForModule3ServerInterfaces = tcForModule3.OPCUAServerInterfaceFolderPath;

// Retrieve the OPC UA server interface selected for a “tcForModule3”

ServerInterfcaes tcForModule3ServerInterfaceType = tcForModule3.OPCUAServerInterfaceType;

//Setting the scope of a “tcForModule4”
tcForModule4.SetScope(tcForModule3OPCUAServeraddress, tcForModule3ServerInterfaceType,
tcForModule3ServerInterfaces);
```

Error message during scope modification

The following user scenarios and corresponding error messages are tabulated below.

User Scenarios	Error messages
“OPCUAServerAddress” is not mentioned in right format i.e.,opc.tcp://server:port/path	Specified 'OPCUAServerAddress' is invalid. Please use the format 'opc.tcp://server:port/path'
When null value is used for “OPCUAServerAddress”	'OPCUAServerAddress' cannot be null or empty. Please use the format 'opc.tcp://server:port/path'
Trying to update the scope of a test case in secondary project.	Error message should be from openness framework
No matching files exists in the specified “OPCUAServerInterfaceFolderPath”	Specified 'OPCUAServerInterfaceFolderPath ' does not contain valid server interface files
Folder path mentioned at “OPCUAServerInterfaceFolderPath” is not available	Cannot find directory '<Drive_letter>:\\'
Folder path mentioned at “OPCUAServerInterfaceFolderPath” contains illegal characters	Illegal characters in path
Folder path mentioned at “OPCUAServerInterfaceFolderPath” is unauthorized path	Cannot read file '<Folderpath>'
Folder path mentioned at “OPCUAServerInterfaceFolderPath” is Specific/relative path	The argument ‘<path>’ cannot be a specific path
Folder path mentioned at “OPCUAServerInterfaceFolderPath” contains additional whitespace characters (i.e. leading and trailing whitespace)	The argument '<path>' contains additional white-space characters which is not allowed.
Folder path mentioned at “OPCUAServerInterfaceFolderPath” from a disconnectable/removable path	Cannot find directory 'G:\\' for path 'G:\\RemovableUSB_Disk'
Folder path mentioned at “OPCUAServerInterfaceFolderPath” contains more than 248 characters	The specified path, file name, or both are too long.

5.19.4.7 Execution of System Test Cases

Execution of System Test Cases

Test case execution can be performed in three ways in openness application as mentioned below.

1. On specific test case.
2. On list of test cases.
3. All available test cases in System test folder.

Reference

Signature

```
public SystemTestCaseExecutor GetService<SystemTestCaseExecutor>();
public TestResults Run(SystemTestCase myTestCase);
public TestResults Run(new IEnumerable<SystemTestCase>() {myTestCase1,myTestCase2,.. });
public TestResults Run(SystemTestGroup allTestCases);
```

The following attributes are supported by TestResults:

Attribute name	Data type	Access mode	Description
Messages	Siemens.Engineering.Test-Suite.TestResults-MessgeComposition	Read	Composition of feedback message
Parent	IEngineeringObject	Read	Returns the parent ofthe container
State	Siemens.Engineering.Test-Suite.TestResults-State	Read	Final state in a test result feedback message with possibly values:Warning, Error and Success
Warning-Count	Int	Read	Number of warnings in a test result feedback message
ErrorCount	Int	Read	Number of errors in atest result feedback message
Description	String	Read	Description or contentof a test result feedback message
DateTime	DateTime	Read	Date and Time in a test result feedback message
Path	String	Read	Path to a test result feedback message

Methods Used:

Name of the method	Description
Run()	To run the selected test case(s)

Note

Time Stamp Format

All time stamps are in UTC. If you want to see the local time you can `DateTime.ToLocalTime()`.

Program Code

You can modify and use the following program code example:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
SystemTestCaseExecutor systemTestService =
testSuite.SystemTestGroup.GetService<SystemTestCaseExecutor>();
//To execute specific test case:
//Retrieve single test case:
//Test case at 0th index is available in the project:
SystemTestCase myTestCase1 = testSuite.SystemTestGroup.SystemTestCases[0];
//TestCase_10 is available in the project:
SystemTestCase myTestCase2 = testSuite.SystemTestGroup.SystemTestCases.Find("TestCase_10");
//Execute above retrieved test cases:TestResults TestCaseResults1 =
systemTestService.Run(myTestCase1);
TestResults TestCaseResults2 = systemTestService.Run(myTestCase2);
//To execute list of test cases://Retrieve list of test cases to execute:
IEnumerable<SystemTestCase> TestCaseList = new List<SystemTestCase>()
{ myTestCase1,myTestCase2 };
//Execute above retrieved list of test cases:
TestResults TestCasesResults3 = systemTestService.Run(TestCaseList);
//To execute all test cases in SystemTestGroup:
//Retrieve list of test cases to execute:
SystemTestSystemGroup systemTestFolder = testSuite.SystemTestGroup;
//Execute all available test cases in SystemTestFolder:
TestResults TestCasesResults4 = systemTestService.Run(systemTestFolder);
```

User can access above test results and iterate through the TestResults messages using the following code:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
...
//Retrieve test results of single test case execution
WriteTestResults(TestCaseResults1);
//Retrieve test results of list of test cases execution
WriteTestResults(TestCasesResults3);
//Retrieve test results of test cases execution available in "SystemTestGroup"
WriteTestResults(TestCasesResults4);
...
private void WriteTestResults (TestResults result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);
RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(TestResultsMessageComposition messages,string
indent= "")
{
indent += "\t";
foreach (TestResultsMessage message in messages)
{
Console.WriteLine(indent + "Path: " + message.Path);
Console.WriteLine(indent + "DateTime: " + message.DateTime);
Console.WriteLine(indent + "State: " + message.State);
Console.WriteLine(indent + "Description: " + message.Description);
Console.WriteLine(indent + "Warning Count: " + message.WarningCount);
Console.WriteLine(indent + "Error Count: " + message.ErrorCount);
RecursivelyWriteMessages(message.Messages, indent);}}
```

5.19.4.8 Import/Export of system test cases using openness APIs

Methods used

The following methods are used to import and export the test cases:

Name of the method	Description
`SaveToFile()`	To export the system test case
`LoadFromFile()`	To import the system test case(s)

Saving a test case to an external textual file

Requirement

The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `SaveToFile()` method on specific test case to export it to a selected path.

Signature

```
public void SaveToFile(FileInfo path);
```

The following table shows the needed method parameters:

Name of the parameter	Description
FileInfo	Full Path for the system test case to be saved

Program Code

Here, `motorFBTestCase` and `motorFBTestCase1` are test cases available in "MyProject".

```
motorFBTestCase.SaveToFile(exportFile);
motorFBTestCase1.SaveToFile(new FileInfo(@"D:\Temp\TSTestcases2.tst"));
```

Loading test cases into project from external textual file

Requirement

The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `LoadFromFile()` method on the `SystemTestCaseComposition` to import a test case(s) from the selected file.

Signature

```
public IList<SystemTestCase> LoadFromFile (FileInfo
path,ImportOptions loadOptions, TCLoadOptions tcLoadOptions);
```

The following table shows the needed method parameters:

Name of the parameter	Description
`FileInfo`	Full Path for the test case to be saved
`ImportOptions.None`	Throws exception if the test case with the same name exists in the project
`ImportOptions.Override`	Overwrites existing test case in the project with the same name
`tcLoadOptions.IgnoreInvalidObject`	Imports the test case though the test case scope(OPC UA server address and interface selected /properties in the file is invalid/missing in the file)
`tcLoadOptions.None`	Throws exception if test case imports with invalid/missing OPC UA server address and also if file contains Interface_Selected/ test case properties are invalid.

Program code

Here, motorFBTestCases is an instance of a SystemTestCaseComposition.

Examples:

```
// Load test cases from above obtained file path to project
// Throw Exception if the test case with the same name exists already
// Import the test case with a invalid scope/properties
FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tst");
motorFBTestcases.LoadFromFile(LoadFile,
ImportOptions.None,TCLoadOptions.IgnoreInvalidObject);
// Load test cases from above obtained file path to project
// Throw Exception if the test case with the same name exists already
// Throw Exception if the test case created with a invalid/not available scope/invalid
properties
FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tst");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.None, TCLoadOptions.None);
// Load test cases from above obtained file path to project
// Import test case with the same name exists already
// Import the test case with a invalid scope/properties
FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tst");
motorFBTestcases.LoadFromFile(LoadFile,
ImportOptions.Override,TCLoadOptions.IgnoreInvalidObject);
// Load test cases from above obtained file path to project
// Overwrite test case with the same name exists already in project
// Throw Exception if the test case created with a invalid/not available scope/invalid
properties
FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tst");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.Override, TCLoadOptions.None);
```

5.19.4.9 Working with Libraries

Copy System Test Case to Libraries

System test cases from project can be copied to the libraries for storage and can be reused and modified. You do not have to explicitly export/import system test cases for external modification.

If the project library is opened in read-write mode, you can create a MasterCopy of an IMasterCopySource at target location.

Reference

The following table describes the syntax of the class:

```
public MasterCopy Create (IMasterCopySource sourceObject);
```

Methods Used

Name of the method	Description
`Create()`	To copy system test case from project to Project/ Global libraries

Program Code

You can modify and use the following program code example to copy test case from project to libraries:

```
using Siemens.Engineering;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
...

// Retrieving a system test case to copy to libraries:
SystemTestCase mySystemTestCase =
testSuite.SystemTestGroup.SystemTestCases.Find("CalcTestCase");

// For copying a system test case to project library master copy system
folder:
MasterCopy masterCopyInSystemFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Create(mySystemTes
tCase );

// For copying a system test case to project library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0].MasterCopies.Create(
mySystemTestCase );
tiaPortal.GlobalLibraries.Open(new FileInfo((@"D:\Documents\.."),
OpenMode.ReadWrite);

// For copying a test case to global library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies.Create(mySystem
TestCase );

// For copying a test case to global library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies.Crea
te(mySystemTestCase );
```

Copy Master copies to projects

The TIA Portal Openness API interface supports copying of master copies to project using the `CreateFrom` action. The action will create a new object based on the source master copy and place it in the composition where the action was called. The action will try to create the new system test case/s with the same name as the source master copy. If such name is available in SystemTestCaseComposition, the system will give the new test case a new name. Then, it will return the new system test case copied.

Reference

The following table describes the syntax of the class:

```
public SystemTestCase CreateFrom (MasterCopy masterCopy);
```

Methods used

Name of the method	Description
`CreateFrom()`	To copy master copies from Project/Global libraries to project

Program Code

You can modify and use the following program code example to copying master copies from project to libraries:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;

...

// For accessing master copy object in global library to copy to project:
var masterCopyObjectinGL =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinGL =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies[1];

// To copy system test case from global library system folder to project:
SystemTestCase SystemtestCase1
=testSuite.SystemTestGroup.SystemTestCases.CreateFrom(masterCopyObjectinGL
);

// To copy test case from global library user folder to project:
SystemTestCase SystemtestCase2
=testSuite.SystemTestGroup.SystemTestCases.CreateFrom(masterCopyObjectinUs
erFolderinGL);

// For accessing master copy object in project library to copy to project:
var masterCopyObjectinPL
=tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinPL
=tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0].MasterCopies[0];

// To copy system test case from project library system folder to project:
SystemTestCase SystemtestCase3
=testSuite.SystemTestGroup.SystemTestCases.CreateFrom(masterCopyObjectinPL
);

// To copy systemtest case from project library user folder to project:
SystemTestCase SystemtestCase4
=testSuite.SystemTestGroup.SystemTestCases.CreateFrom(masterCopyObjectinUs
erFolderinPL);
```

Copy Master copies (of type SystemTestCase) within and between libraries

To copy master copies (of type SystemTestCase) within and between libraries, Refer to **TIA Portal Openness API** under chapter **Funtions on libraries** in the topic **Copying master copies** from the Information system.

5.19.5 Exceptions

Handling exceptions

Exceptions when accessing the Test Suite via TIA Portal Openness API

During the execution of a Test Suite application via the TIA Portal Openness API, all errors which occur are reported as exceptions. These exceptions contain information that will help you to correct the errors which have occurred.

A distinction is made between two types of exceptions:

- Recoverable (Siemens.Engineering.EngineeringException)
  You can continue to access the TIA Portal without interruption with this exception. Alternatively, you can cancel the connection to the TIA Portal.
  The EngineeringExceptions include the following types:
  - Exceptions when executing test case (SupportSimulationNotEnabledException), when the support simulation property is not enabled in the project.
  - OPCUA Server invalid address exception (OPCUAServerAddressNotValidException) - This exception occurs when the executed test cases have and invalid OPCUA server address.
  - License not found exception (LicenseNotFoundException) - This exception occurs when a valid Test Suite license in not available
- NonRecoverable (Siemens.Engineering.NonRecoverableException)
  This exception closes the TIA Portal and the connection to the TIA Portal is disconnected. You need to restart the TIA Portal using the TIA Portal Openness application.

5.20 Functions for SINUMERIK 840D sl

5.20.1 Introduction

Using TIA Portal-Openness, you automate the engineering and control the TIA Portal using a program that you created yourself.

In this help document, you can find a lot of information and code examples for this program that you generate yourself. You can also generate and use your own programs for the TIA Portal "SINUMERIK" application.

Further information

Before you generate your own program for SINUMERIK from the sample codes listed in the following, please note the general information on Openness, which you can find in this help under the following keywords:

- Preconditions for using TIA Portal Openness
- Installing TIA Portal Openness
- Access the TIA Portal

- TIA Portal Openness object model
- Programming steps

5.20.2 Type identifier - identifier of the components

Each SINUMERIK component has a unique number, which is designated as type identifier (TypeIdentifier) . In the Openness program code, you can use the type identifier to clearly identify and designate a component. For example, the type identifier for SINUMERIK 840D sl NCU 710.3 PN is "Article No. :6FC5 371-0AA30-0Axx/Vy.z".

The type identifier is displayed when creating a device using the dialog "Add new device" and in the tabular device overview.

You can copy the type identifier into your Openness application.

The type identifier is expected in Openness as a current parameter when a method is called, e.g. the CreateWithItem() method.

Activating display of the type identifier in SINUMERIK

1. In the project view, choose "Options > Settings" from the menu.
   The "Settings" configuration area opens.
2. In the secondary navigation, select entry "Hardware configuration".
3. Activate option "Activate display of the type identifier for devices and modules".
   The type identifier display is now active.

5.20.3 Fundamentals

Overview

You can create the following SINUMERIK devices in the TIA Portal Openness:

- NCU
- NX module
- ADI4

To create the SINUMERIK devices, use the `CreateWithItem()` method of the `Devices Collection`.

Note

All integrated subcomponents of a SINUMERIK-NCU - such as PLC, NCK, CP, HMI and SINAMICS Integrated - are automatically created at the same subordinate level.

Special issue when creating SINUMERIK devices

The names of the subracks correspond to the NCU types and are write protected in the user interface. You must also use the designations of the subracks in the TIA Portal Openness.

Use the following standard names of the subracks for creating a device using a device element type identifier with the `Device CreateWithItem` method:

SINUMERIK NCU	Subrack standard name
SINUMERIK 840D sl NCU 710.3 PN	NCU 710.3 PN
SINUMERIK 840D sl NCU 720.3 PN	NCU 720.3 PN
SINUMERIK 840D sl NCU 730.3 PN	NCU 730.3 PN
SINUMERIK 840D sl NCU 730.3 PN /319	NCU 730.3 PN /319

Note

Alternatively, you can omit the parameter names. The default name is used if the name is "Null" or "String.Empty".

The following parameters can be used with the `CreateWithItem()` method:

- `default name`,
  e.g.
  ```
  project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AA0/
  V4.8", "NCU 710.3 PN", "TestDevice");
  ```
- `null`,
  e.g.
  ```
  project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AA0/
  V4.8", null, "TestDevice");
  ```
- `string.Empty`,
  e.g.
  ```
  project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AA0/
  V4.8", string.Empty, "TestDevice");
  ```

More information about the call parameters associated with the `CreateWithItem()` method is provided in Chapter "Creating a device".

The following table lists the assignment of the devices and their type identifiers:

SINUMERIK NCU	Type identifier
SINUMERIK 840D sl NCU 710.3 PN	Article No. :6FC5 371-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 720.3 PN	Article No. :6FC5 372-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 730.3 PN	Article No. :6FC5 373-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 730.3 PN /319	Article No. :6FC5 373-0AA31-0Axx/Vy.z

When creating SINUMERIK devices, place holders in the type identifier are permissible. You can subsequently replace these placeholders by device-specific characters. If you enter type identifier "OrderNumber:6FC5 372-0***0-0AA0/V4.8" for example, then SINUMERIK 840D sl NCU 720.3 PN is created with firmware version FW4.8.

Classification of the device element

Every device or device element has obligatory attributes, which are read and written to. More information about the attributes that are supported in Openness is provided in Chapter "Functions on device elements".

The classification attributes of the device element are write protected, and not visible in the user interface of the TIA Portal.

The classification attributes have the value "DeviceItemClassification":

Value of the classification attribute	Description
DeviceItemClassifications.None (0)	No classification.
DeviceItemClassifications.CPU (1)	The device element is a CPU.
DeviceItemClassifications.HM (2)	The device element is a header module.

If you interrogate the value of the device element via the integrated PLC, then for a SINUMERIK device, this is the value "CPU (1)".

In the Openness object model, the following components act as header modules:

- SINAMICS Integrated
- NX module
- ADI4

In all other cases, the value of the classification attribute "`DeviceItemClassification`" is "`None`" (0).

Finding device elements via the "Header module" property

The following example shows how you can find device elements with the "Header module" property, before configuring a telegram, for example.

You can only read this property, but cannot set it at device elements.

Finding device elements via the "Header module" property

```
foreach (Device device in project.Devices)
{
  foreach (DeviceItem deviceItem in device.DeviceItems)
   {
    if (deviceItem.Classification == DeviceItemClassifications.HM)
     {
      var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
      // do something
     }
   }
}
```

See also

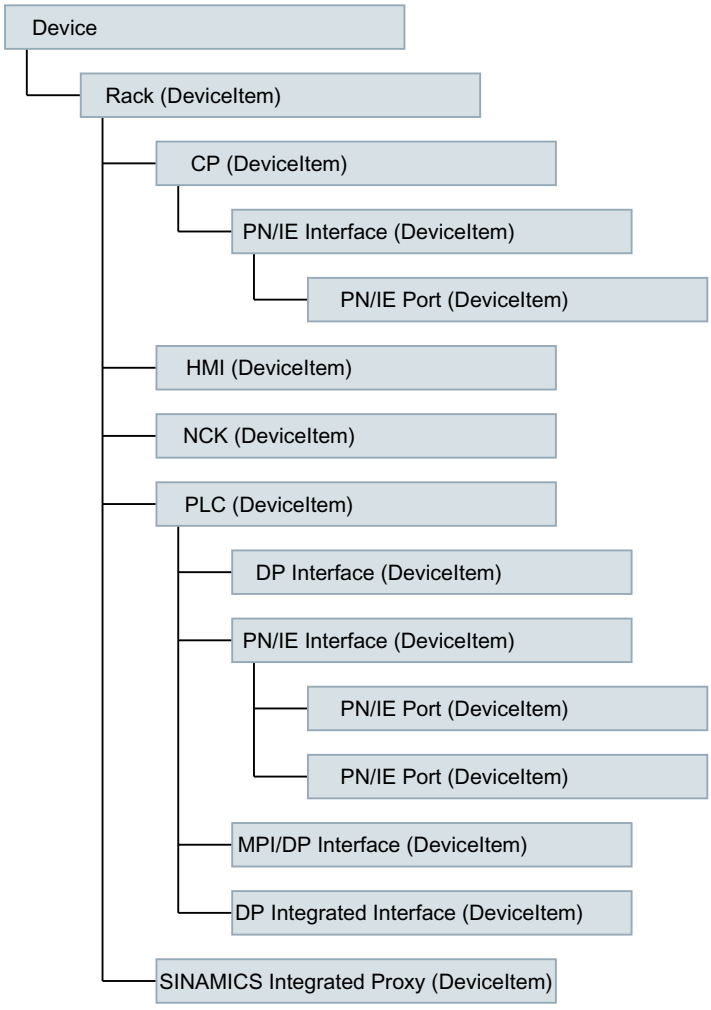
Creating an NX module (Page 1042)

Creating an NCU (Page 1041)

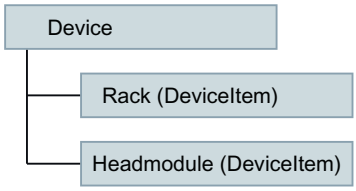
5.20.4 Object model

Overview

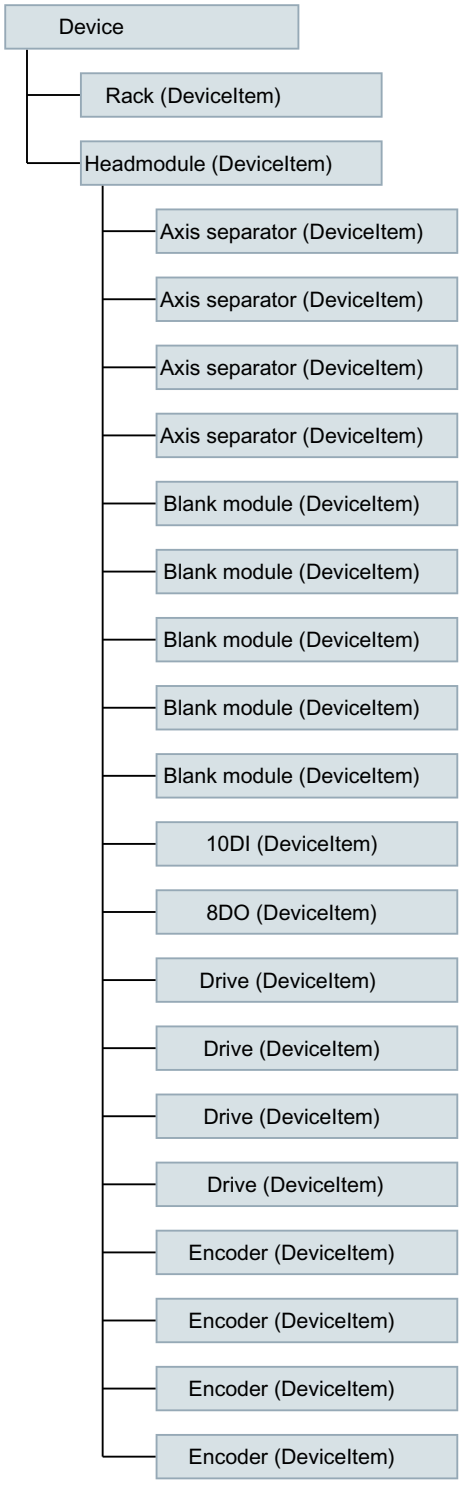
The following diagram describes the objects located under "Device" for SINUMERIK 840D sl:



The following diagram describes the objects located under "Device" for SINUMERIK NX 10.3/15.3:



The following diagram describes the objects located under "Device" for ADI4:



For more information on TIA Portal Openness object model, see Chapter "TIA Portal Openness APII".

5.20.5 Code example

5.20.5.1 General

The following code examples describe the basic procedure for various applications. The code is not necessarily complete or compilable.

5.20.5.2 Executing the first steps in SINUMERIK

- Connect the TIA Portal Openness application with the TIA Portal.
- Open your project.

The following example shows how you can determine which SINUMERIK Toolbox version is installed.

Determining the SINUMERIK Toolbox version

```
using Siemens.Engineering;
if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINUMERK Toolbox") &&
sw.Version.Equals("V17")))
 {
  Console.WriteLine("SINUMERIK Toolbox is available");
 }
  // "V17" is the current SINUMERIK version started at October 2021.
```

5.20.5.3 Creating an NCU

You create a SINUMERIK NCU using the `CreateWithItem()` method of the `Devices Collection`. SINUMERIK NCUs are specified using method parameters. The format of the parameters is described in the following.

Creating an NCU

Parameter format for SINUMERIK NCU:

```
CreateWithItem(@"OrderNumber:mlfb/
FirmwareVersion/","NameOfTheDevice",positionNumber)
```

The `positionNumber` parameter is optional.

The following example shows how you create a SINUMERIK 840D sl NCU 720.3 PN control system.

Creating a SINUMERIK 840D sl NCU 720.3 PN

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open("..."); //The path of the project

Device NCUDevice = tiaproject.Devices.CreateWithItem("@OrderNumber:6FC5
372-0AA30-0AA0/4.8/", "NCU 720.3 PN", string.Empty, "TestDevice");
```

5.20.5.4 Creating an NX module

You create an NX module using the `Device CreateWithItem` method. You then connect the NX module to an NCU.

The following type identifiers are used for SINUMERIK NX modules:

SINUMERIK NX module	Type identifier
SINUMERIK NX10.3	OrderNumber:6SL3 040-1NC00-0Axx/Vy.z
SINUMERIK NX15.3	OrderNumber:6SL3 040-1NB00-0Axx/Vy.z

The following example shows how you can create a SINUMERIK NX module.

Creating an NX module

```
project.Devices.CreateWithItem("OrderNumber:6SL3040-1NC00-0AA0/
V5.1", "MyNXDevice", "TestDevice");
```

Version compatibility

The firmware version of the NX module must match the firmware version of SINAMICS Integrated - and must be compatible with the firmware version of the NCU.

An NX firmware version that deviates from SINAMICS Integrated cannot be assigned via Openness.

The following table lists the version compatibility for 840D sl:

NCU firmware (840D sl)	SINAMICS Integrated/NX firmware
V4.95	V5.2

5.20.5.5 Connecting an NX module with NCU

Connecting an NX module with an NCU

To connect an NX module with an NCU via subnet type "`ProfibusIntegrated`", the "`NetworkInterface`" service must be loaded via the header module of NX.

The following example shows how you can load the "`NetworkInterface`" service.

Loading the NetworkInterface service

```
foreach (Device device in project.Devices)
{
  foreach (DeviceItem deviceItem in device.DeviceItems)
  {
   if (deviceItem.Classification == DeviceItemClassifications.HM)
   {
    var networkInterface = deviceItem.GetService<NetworkInterface>();
    // do something
   }
  }
}
```

The following example shows how you can connect an NX module with an NCU via "ProfibusIntegrated":

Connecting an NX module via ProfibusIntegrated

```
Subnet pbiSubnet = ...;
Node node = networkInterface.Nodes.FirstOrDefault();
node.ConnectToSubnet(pbiSubnet);
```

The DP address is assigned to the fixed NX in Openness via DRIVE-CLiQ port. Each port label has a fixed DP-integrated address.

The DP address must be assigned to the NX before connection to the NCU.

5.20.5.6 Activating Safety Integrated

Activating Safety Integrated

With TIA Portal Openness, you can activate Safety Integrated (F-PLC) in the properties of the NCU.

Note

Effects on telegram configuration

The Safety Integrated mode used affects the telegram configuration because telegrams other than the ones used for inactive Safety Integrated mode are used in Safety Integrated plus (F-PLC) mode.

However, added or changed telegrams are retained if they are compatible with the newly selected Safety Integrated mode.

If applicable, after the mode change in the telegram configuration, ensure that any adjustments are still available.

You activate and deactivate Safety Integrated (F-PLC) with the `SafetyModeProvider` **service.**

Note

The PLC must be in offline mode if Safety Integrated (F-PLC) is being activated or deactivated.

Note

SafetyModeProvider is included in Namespace Siemens.Engineering.MC.Sinumerik .

The following example shows how to call the `SafetyModeProvider` service:

Calling SafetyModeProvider

```
...
Siemens.Engineering.HW.Device cnc = ...;
try
{
   SafetyModeProvider provider = cnc.GetService<SafetyModeProvider>();
   //Perform the safety mode change:
   provider.SetSafetyMode(SafetyMode.DbSI);
}
catch( (EngineeringException ex) )
  {
  // Handle safety mode change failure
  }
```

The following example shows how to call the current Safety Integrated setting on the device:

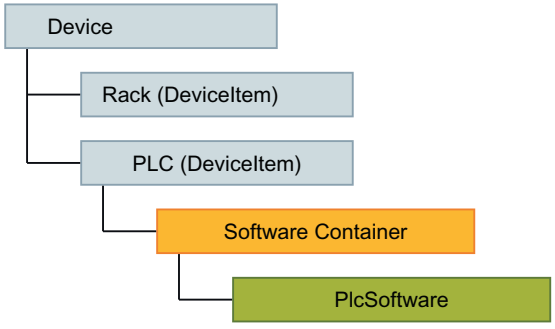
Calling the safety setting of the device

```
...
Siemens.Engineering.HW.Device cnc = ...;
try
{
   SafetyModeProvider provider = cnc.GetService<SafetyModeProvider>();
   //Query the safety mode:
   SafetyMode safetyMode = provider.CurrentMode;
}
catch( (EngineeringException ex) )
{
// Handle any failure
  }
```

5.20.5.7 Accessing PLC software

Overview

The following diagram shows the software container and the PLC software in the object model (Page 1049):



The following code example shows how to find a PLC, irrespective of its specific implementation (integrated SINUMERIK PLC, SIMATIC PLC, software PLC in the PC) based on the "CPU" property:

Finding PLCs

```
Device ncuDevice = ...
DeviceItem plc = GetPlc(ncuDevice.DeviceItems);

   ...
DeviceItem GetPlc(DeviceItemComposition deviceItems)
{
   if (deviceItems.Count == 0)
   {
       return null;
   }
   foreach (var deviceItem in deviceItems)
   {
       if (deviceItem.Classification == DeviceItemClassifications.CPU)
           return deviceItem;

       DeviceItem plc = GetPlc(deviceItem.DeviceItems);
       if (plc != null)
           return plc;
   }

   return null;
}
```

Find out more about accessing the PLC software container under "Accessing software goals".

5.21 Functions for SINUMERIK ONE

5.21.1 Introduction

Using TIA Portal-Openness, you automate the engineering and control the TIA Portal using a program that you created yourself.

In this help document, you can find a lot of information and code examples for this program that you generate yourself. You can also generate and use your own programs for the TIA Portal "SINUMERIK" application.

Further information

Before you generate your own program for SINUMERIK from the sample codes listed in the following, please note the general information on Openness, which you can find in this help under the following keywords:

- Preconditions for using TIA Portal Openness
- Installing TIA Portal Openness
- Access the TIA Portal

- TIA Portal Openness object model
- Programming steps

5.21.2 Type identifier designation of the components

Each SINUMERIK component has a unique number, which is designated as type identifier (TypeIdentifier) . In the Openness program code, you can use the type identifier to clearly identify and designate a component. For example, the type identifier for SINUMERIK ONE NCU 1750 "Article No.: 6FC5 317-5AA00-0Axx/Vy.z".

The type identifier is displayed when creating a device using the dialog "Add new device" and in the tabular device overview.

You can copy the type identifier into your Openness application.

The type identifier is expected in Openness as a current parameter when a method is called, e.g. the CreateWithItem() method.

Activating display of the type identifier in SINUMERIK

1. In the project view, choose "Options > Settings" from the menu.
   The "Settings" configuration area opens.
2. In the secondary navigation, select entry "Hardware configuration".
3. Activate option "Activate display of the type identifier for devices and modules".
   The type identifier display is now active.

5.21.3 Fundamentals

You can create the following SINUMERIK devices in the TIA Portal with Openness:

- NCU
- NX module
- PPU

To create the SINUMERIK devices, use the method `CreateWithItem()` of the `Devices` Collection.

Note

All integrated subcomponents of a SINUMERIK-NCU - such as PLC, NCK, CP, HMI and SINAMICS Integrated - are automatically created at the same subordinate level.

Special issues when creating SINUMERIK devices

The names of the subracks correspond to the NCU types and are write protected in the user interface. You must also use the names of the subracks in the TIA Portal Openness. Use the following standard names of the subracks for creating a device using a device element type identifier with the `Device CreateWithItem` method:

SINUMERIK device	Standard subrack names
SINUMERIK ONE NCU 1740	NCU 1740
SINUMERIK ONE NCU 1750	NCU 1750
SINUMERIK ONE NCU 1760	NCU 1760
SINUMERIK ONE PPU 1740	PPU 1740

Note

Alternatively, you can omit the parameter names. The standard name is used if the name is "null" or "String.Empty".

The following parameters can be used with the `CreateWithItem()` method:

- `default name`, e.g. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.15", "NCU 1750", "TestDevice");`
- `null`, e.g. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.15", null, "TestDevice");`
- `string.Empty`, e.g. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.13", string.Empty, "TestDevice");`

Additional information about the call parameters associated with the `CreateWithItem()` method is provided in Chapter "Creating a device".

The following table lists the assignment of the devices and their type identifiers:

SINUMERIK device	Type identifier
SINUMERIK ONE NCU 1740	Article No.: 6FC5 317-4AA00-0Axx/Vy.z
SINUMERIK ONE NCU 1750	Article No.: 6FC5 317-5AA00-0Axx/Vy.z
SINUMERIK ONE NCU 1760	Article No.: 6FC5 317-6AA00-0Axx/Vy.z
SINUMERIK ONE PPU 1740	Article No.: 6FC5 317-4AA00-1xxx/Vy.z

When creating SINUMERIK devices, place holders in the type identifier are permissible. You can subsequently replace these placeholders by device-specific characters.

Classification of the device element

Every device or device element has obligatory attributes, which are read and written to. Additional information about the attributes that are supported in Openness is provided in Chapter "Functions on device elements".

The classification attributes of the device element are write protected, and not visible in the user interface of the TIA Portal.

The classification attributes have the value "DeviceItemClassification":

Value of the classification attribute	Description
DeviceItemClassifications.None (0)	No classification.
DeviceItemClassifications.CPU (1)	The device element is a CPU.
DeviceItemClassifications.HM (2)	The device element is a header module.

If you interrogate the value of the device element via the integrated PLC, then for a SINUMERIK device, this is the value "CPU (1)".

In the Openness object model, the following components act as header module:

- SINAMICS Integrated
- NX module

In all other cases, the value of the classification attribute is "DeviceItemClassification" "None" (0).

Finding device elements via the "Header module" property

The following examples show how you can find device elements with the "Header module" property, for example, before you configure a telegram.

You can only read this property, but cannot set it at device elements.

Finding Sinamics Integrated / NX via the "Header module" property

```
foreach (Device device in project.Devices)
{
  foreach (DeviceItem deviceItem in device.DeviceItems)
   {
    if (deviceItem.Classification == DeviceItemClassifications.HM)
     {
      var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
      // do something
     }
   }
}
```

See also

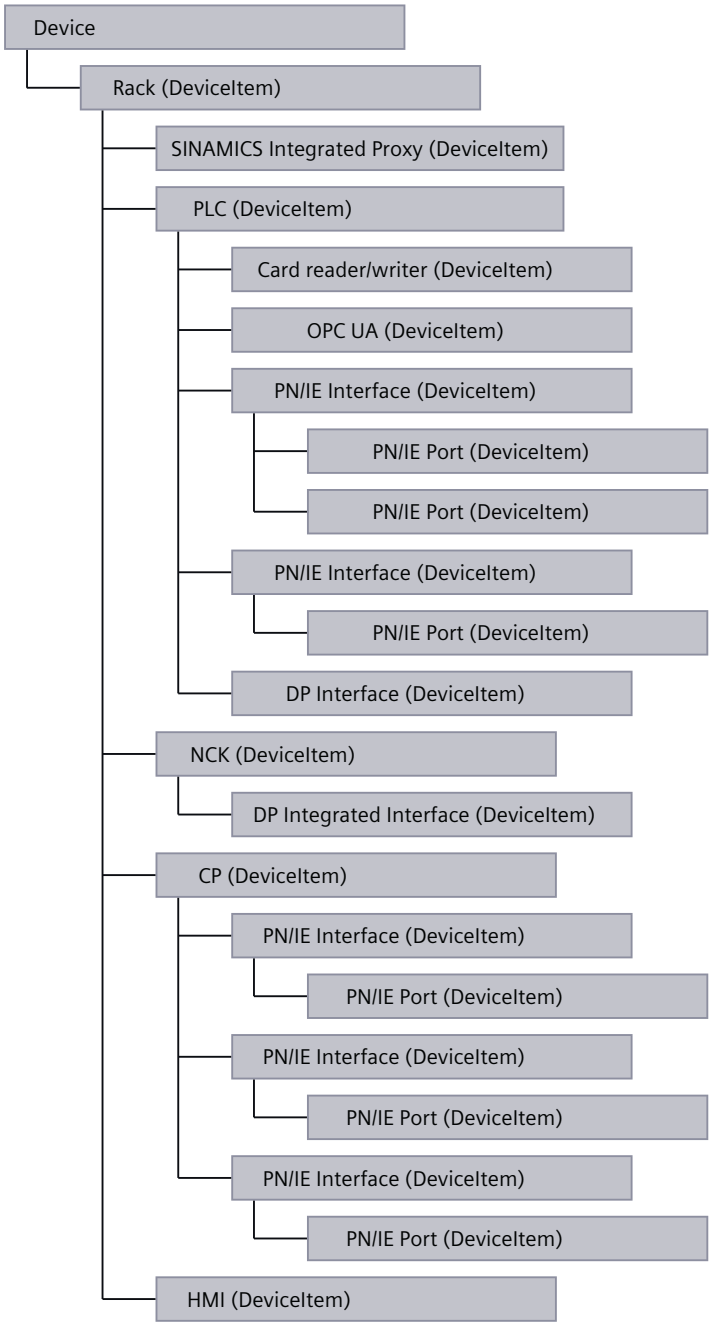
Creating an NCU (Page 1069)

Creating an NX module (Page 1069)

5.21.4 Object model

Overview

The following diagram describes the objects located under "Device" for SINUMERIK ONE:



For more information on the TIA Portal Openness object model, see section "TIA Portal Openness APIl".

Available namespaces for telegram configuration

The following namespaces are valid in SINUMERIK Openness for the telegram configuration interfaces:

Namespace	Assembly
`Siemens.Engineering.MC.DriveConfiguration` (Page 1051)	`Siemens.Engineering.MC.DriveConfiguration` in `Siemens.Engineering.dll`
`Siemens.Engineering.MC.Drives` (Page 1060)	`Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

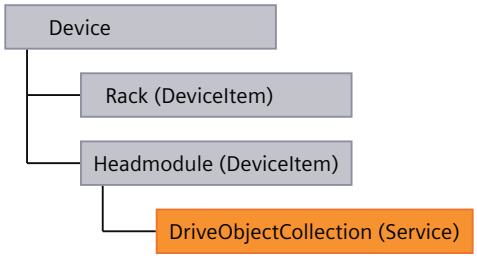
Note

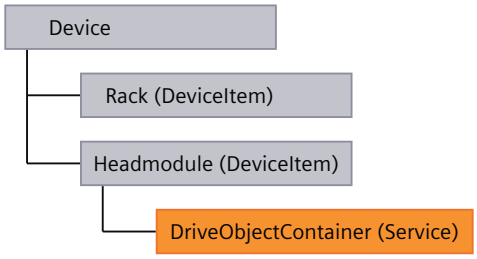
With the namespace `Siemens.Engineering.MC.DriveConfiguration`, additional functions are available, e.g. you can create, delete or reorder drive objects.

In the long term, the namespace `Siemens.Engineering.MC.DriveConfiguration` remains. However, the namespace `Siemens.Engineering.MC.Drives` will still be supported into the next versions of Openness for compatibility reasons.

The following diagrams describe the objects located under "Device" for SINUMERIK NX 10.3/15.3 (availability of services depends on the namespace).

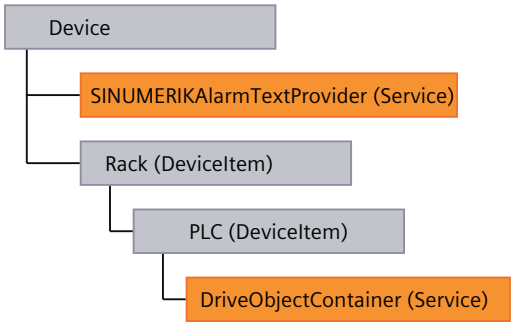
For new applications, use the namespace Siemens.Engineering.MC.DriveConfiguration (and thus the DriveObjectCollection service) because the namespace Siemens.Engineering.MC.Drives (and thus the DriveObjectContainer service) is still supported for compatibility reasons, but offers less functionality.





Importing alarm texts

The following diagram shows the SinumerikAlarmTextProvider object, which is used to import alarm texts **for DB2 alarms**.



In the project navigator, you will find the imported alarms in the following locations:

- in the PLC text lists, e.g. under "CNC_1 > PLC_1 > PLC Text lists"
- in the project texts, e.g. under "Project_1 > Common data > Project texts".

More information and code examples can be found at Importing SINUMERIK PLC alarm texts (Page 1076).

Information on importing SINUMERIK user alarms (in user text lists) can be found in the TIA Portal help under Exporting/importing PLC message text lists.

5.21.5 Reference

5.21.5.1 Namespace Siemens.Engineering.MC.DriveConfiguration

DriveObject

DriveObject

The `DriveObject` class allows access to the drive object. Using the drive object, the telegram can be accessed, for example.

Namespace: `Siemens.Engineering.MC.DriveConfiguration`
Assembly: `Siemens.Engineering.MC.DriveConfiguration` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
Telegrams	TelegramComposition	read	-	Returns a list with the available telegrams of the drive object. The list can be changed with the TelegramComposition class.
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (DriveObjectCollection and DriveObjectContainer).
Category		read	-	Returns the reference to the higher-level class (DriveObjectCategory)
Name		read	-	Returns the name of the drive object.

The following table describes the **methods** of the class:

Name	Description
GetAttribute	Performs read access to an attribute of a drive object
GetAttributes	Performs read access to all attributes of a drive object
GetAttributeInfos	Performs read access to information on attributes of a drive object
SetAttribute	Performs write access to an attribute of a drive object
SetAttributes	Performs write access to all attributes of a drive object
Delete	Deletes the drive object instance at which the "Delete" is called

DriveObjectCollection

DriveObjectCollection

The DriveObjectCollection is a service of the drive object (DeviceItem) of the current device (Device).

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration in Siemens.Engineering.dll

The following table describes the **navigators** of the DriveObjectCollection:

Name	Data type	Access mode	Access	Description
DriveObjects	DriveObjectComposition	read	-	Returns a list with the available drive objects. The drive objects allow access to the drive parameters and telegrams.

The following table describes the **methods** of the class:

Name	Description
GetAttribute	Performs read access to an attribute of DriveObjectCollection
GetAttributes	Performs read access to all attributes of DriveObjectCollection
GetAttributeInfos	Performs read access to information on attributes of DriveObjectCollection
SetAttribute	Performs write access to an attribute of DriveObjectCollection
SetAttributes	Performs write access to all attributes of DriveObjectCollection

DriveObjectComposition

DriveObjectComposition

The DriveObjectComposition class allows access to available telegrams of a drive object and contains all of the drive objects of an NCU or NX module.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (DriveObjectContainer).
Count		read	-	
IsReadOnly		read	-	

The following table describes the **methods** of the class:

Name	Description
GetEnumerator	Facilitates the iteration over the specified set of elements
Contains	Determines whether the specified instance is contained in the set of the elements.TRUE: The container contains the instance FALSE: The container does not contain the instance
IndexOf	Returns the index in the set of the elements for the queried instance.
Create	Creates a DriveObjectComposition class
Find	Searches for a DriveObjectComposition class

DriveObjectCategory

DriveObjectCategory

The enum `DriveObjectCategory` contains predefined categories of the drive objects. Read access to `DriveObjectCategory` is possible.

Namespace: `Siemens.Engineering.MC.DriveConfiguration`
Assembly: `Siemens.Engineering.MC.DriveConfiguration` in `Siemens.Engineering.dll`

The following table contains the predefined categories of the drive objects:

Supported drive objects in SINUMERIK	DriveObjectCategory
`SinumerikDriveAxis`	`SERVO`, `ENC`, `HLA`, `TM41`
`SinumerikControlUnit`	`CU_I`, `CU_NX_CX`
`SinumerikInfeed`	`A_INF`, `B_INF`, `S_INF`

TelegramComposition

TelegramComposition

The `TelegramComposition` class facilitates the access to the telegrams of a drive object (DriveObject (Page 1051)). The structure of a telegram can be read out via the Telegram (Page 1055) class.

Changes to telegram objects (e.g. changing the safety telegram) can result in the relevant telegram object being deleted in the `TelegramComposition` or a new telegram object being created. In this case, you must search through the `TelegramComposition` again to find the new telegram object (Page 1055) again after the change.

If the respective drive object does not support telegrams, the value for `TelegramComposition` is returned blank.

Namespace: `Siemens.Engineering.MC.DriveConfiguration`
Assembly: `Siemens.Engineering.MC.DriveConfiguration` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
`Parent`	`IEngineeringObject`	`read`	-	Returns the reference to the higher-level class (`DriveObject`).
`Count`		`read`	-	
`IsReadOnly`		`read`	-	

The following table describes the **methods** of the class:

Name	Description
`Create(enum TelegramId Id)`	Creates a telegram; the type of the telegram corresponds to the ID.
`int IndexOf(TelegramType)`	Returns the index in the set of the elements for the queried instance.
`bool Contains`	Determines as to whether the specified instance is included in the set of elements. `TRUE`: The container contains the instance `FALSE`: The container does not contain the instance
`IEnumerator GetEnumerator`	Allows `IEnumerator<DriveObject>` the iteration via the given set of the elements

Telegram

Telegram

The `Telegram` class allows access to the structure of a telegram from a drive object.

Namespace: `Siemens.Engineering.MC.DriveConfiguration`
Assembly: `Siemens.Engineering.MC.DriveConfiguration` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
`Id`	`Int32`	read	-	Returns the ID of the telegram.
`Type`	`enum`: TelegramType (Page 1056)	read	-	Returns the type of telegram as `Enum TelegramType`.
`Addresses`	AddressComposition (Page 1058)	read	-	Returns an `AdressComposition` with information on the address.

The following table describes the **methods** of the class:

Name	Description
`GetSize(`AddressIoType (Page 1059)`)`	Returns the size of the inputs or outputs of the telegram.
`CanSetSize (`AddressIoType (Page 1059)`, Int32, bool)`	Returns `true` if the size of the telegram can be changed as parameterized.
`SetSize`	Performs write access to the telegram size
`Delete`	Deletes the telegram instance at which "`Delete`" is called
`GetAttribute`	Performs read access to a telegram attribute
`GetAttributes`	Performs read access to all telegram attributes
`GetAttributeInfos`	Performs read access to information on telegram attributes

Name	Description
SetAttribute	Performs write access to a telegram attribute
SetAttributes	Performs write access to all telegram attributes
Delete	Deletes the telegram

Properties of Safety telegrams

The following table describes the additional **properties** of the "Telegram" class of the type "SafetyTelegram".

Name	Data type	Access mode	Access	Description
Failsafe_FSourceAddress	UInt32	read	Dynamic	PROFIsafe source address
Failsafe_FDestinationAddress	UInt32	read/ write	Dynamic	PROFIsafe destination address
Failsafe_FIODBNumber	UInt32	read/ write	Dynamic	Safety DB number, write access only possible if the property "Failsafe_ManualAssignmentFIODBNumber" is assigned the value "1"
Failsafe_FIODBName	string	read	Dynamic	Name Safety DB
Failsafe_ManualAssignmentFIODBNumber	bool	read/ write	Dynamic	Setting for manual assignment of the property "Failsafe_FIODBNumber"
Failsafe_FMonitoringtime	UInt32	read/ write	Dynamic	Safety monitoring time, write access only possible if the property "Failsafe_ManualAssignmentFMonitoringtime is set to "true"
Failsafe_ManualAssignmentFMonitoringtime	bool	read/ write	Dynamic	Setting for manual assignment of the property "Failsafe_FMonitoringtime"

The additional properties are accessed via GetAttribute and SetAttribute, e.g. GetAttribute("Failsafe_FSourceAddress").Uint32 is expected as return value.

TelegramType

TelegramType

The Enum TelegramType contains predefined telegram types.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration in Siemens.Engineering.dll

The following table describes the **enum entries**:

Name	Description
MainTelegram	Main telegram: e.g. telegram 136
SupplementaryTelegram	Supplementary telegram: e.g. telegram 701
AdditionalTelegram	Additional telegram: free telegram

Name	Description
SafetyTelegram	Safety telegram: e.g. telegram 903
ProcessMonitoringTelegram	Process monitoring telegram, e.g. telegram 1100

Note

The torque telegram (TorqueTelegram) is not supported, although it is available in the TIA Portal Openness API in the SINUMERIK context.

TelegramPart

TelegramPart

The enum TelegramPart contains the parts of a telegram.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration in Siemens.Engineering.dll

The following table describes the **enum entries**:

Name	Description
Unknown	Unknown part of a telegram
Overall	All parts of a telegram in one telegram: BaseTelegram, Extension and PIV[1)] of a telegram
BaseTelegram	Main part of a telegram
Extension	Telegram extension
PKW	Parameter identification value of a telegram

[1)] PKW is not available in the SINUMERIK context.

TelegramId

TelegramId

The enum TelegramId contains the telegram numbers that are relevant for communication between the PLC and the drive. The IDs are defined via the PROFIdrive standard.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration in Siemens.Engineering.dll

AddressComposition

AddressComposition

The `AddressComposition` class represents the address of a telegram.

namespace: `Siemens.Engineering.MC.DriveConfiguration`
Assembly: `Siemens.Engineering.MC.DriveConfiguration` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
`IoType`	`enum`: AddressIoType (Page 1059)	read	-	Returns information on the type of the address.
`Context`	`enum`: AddressContext (Page 1059)	read	Access only possible via `GetAttribute` or `SetAttribute`	Returns information on the context of the address.
`StartAddress`	`Int32`	read/ write	-	Returns the start address of the telegram or defines the start address.
`Length`	`Int32`	read	-	Returns the length of the telegram.
`IndexOf`	`int32`	read	-	Returns the index in the set of the elements for the queried instance.
`Contains`	`bool`	read	-	Determines as to whether the specified instance is included in the set of elements. `TRUE`: The container contains the instance `FALSE`: The container does not contain the instance
`GetEnumerator`	`IEnumerator<DriveObject>`	read	-	Facilitates the iteration over the specified set of elements

Note

If you want to navigate back to the telegram via Address.Parent, instead of namespace `Siemens.Engineering.MC.Drives.Telegram`, namespace `Siemens.Engineering.MC.DriveConfiguration.Telegram` is called.

You can find further information about TIA Portal Openness libraries under "Standard libraries".

AddressIoType

AddressIoType

The Enum AddressIoType contains information on the type of the address.

Namespace: `Siemens.Engineering.MC.DriveConfiguration`
Assembly: `Siemens.Engineering.MC.DriveConfiguration` in `Siemens.Engineering.dll`

The following table describes the **enum entries**:

Name	Description
`AddressIoType.None`	The IO type cannot be used
`AddressIoType.Input`	Type is an input address
`AddressIoType.Output`	Type is an output address
`AddressIoType.Diagnosis`	Type is a diagnosis address
`AddressIoType.Substitute`	Type is a substitute address

You can find further information about TIA Portal Openness libraries under "Standard libraries".

AddressContext

AddressContext

The Enum AddressContext contains information on the context of the address.

Namespace: `Siemens.Engineering.MC.DriveConfiguration`
Assembly: `Siemens.Engineering.MC.DriveConfiguration` in `Siemens.Engineering.dll`

The following table describes the **enum entries**:

Name	Description
`AddressContext.None`	No context has been found for the address
`AddressContext.Device`	Context is a device address
`AddressContext.Head`	Context is a head address

You can find further information about TIA Portal Openness libraries under "Standard libraries".

5.21.5.2 Namespace Siemens.Engineering.MC.Drives

DriveObject

DriveObject

The `DriveObject` class allows access to the drive object. Using the drive object, the telegram can be accessed, for example.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
`Telegrams`	`TelegramComposition`	read	-	Returns a list with the available telegrams of the drive object. The list can be changed with the `TelegramComposition` class.
`Parent`	`IEngineeringObject`	read	-	Returns the reference to the higher-level class (`DriveObjectContainer`).

The following table describes the **methods** of the class:

Name	Description
`GetAttribute`	Performs read access to an attribute of a drive object
`SetAttribute`	Performs write access to an attribute of a drive object
`GetEnumerator`	Facilitates the iteration over the specified set of elements

See also

TelegramComposition (Page 1061)

DriveObjectContainer

DriveObjectContainer

The `DriveObjectContainer` is a service of the drive object (`DeviceItem`) for the current device (`Device`).

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **navigators** of the `DriveObjectContainer`:

Name	Data type	Access mode	Access	Description
`DriveObjects`	`DriveObjectComposition`	read	-	Returns a list with the available drive objects. The drive objects allow access to the drive parameters and telegrams.
`Parent`	`IEngineeringObject`	read	-	Returns the reference to the higher-level class (`DeviceItem`).

DriveObjectComposition

DriveObjectComposition

The `DriveObjectComposition` class facilitates the access to available telegrams of a drive object.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
`IndexOf`	`int32`	read	-	Returns the index in the set of the elements for the queried instance.
`Contains`	`bool`	read	-	Determines as to whether the specified instance is included in the set of elements. `TRUE`: The container contains the instance `FALSE`: The container does not contain the instance
`GetEnumerator`	`IEnumerator<DriveObject>`	read	-	Facilitates the iteration over the specified set of elements

TelegramComposition

TelegramComposition

The `TelegramComposition` class facilitates the access to the telegrams of a drive object (DriveObject (Page 1060)). The structure of a telegram can be read out via the Telegram (Page 1063) class.

Changes to telegram objects (e.g. changing the safety telegram) can result in the relevant telegram object being deleted in the `TelegramComposition` or a new telegram object

being created. In this case, you must search through the `TelegramComposition` again to find the new telegram object (Page 1063) again after the change.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **methods** of the class:

Name	Description
`CanInsertAdditionalTelegram(Int32, Int32)`	Returns `true` if an extension can be created in accordance with the parameterized sizes (input and output sizes).
`InsertAdditionalTelegram(Int32, Int32)`	Creates an extension for the drive object in accordance with the parameterized sizes and returns `true` if the extension could be inserted. In the event of an error, an `EngineeringTargetInvocationException` is triggered.
`CanInsertSupplementaryTelegram(Int32)`	Returns `true` if a supplementary telegram can be created in accordance with the parameterized telegram number.
`InsertSupplementaryTelegram(Int32)`	Creates the supplementary telegram with the parameterized telegram number and returns `true` if the telegram could be inserted. In the event of an error, an `EngineeringTargetInvocationException` is triggered.
`CanInsertSafetyTelegram(Int32)`	Returns `true` if a safety telegram can be generated corresponding to the parameterized telegram number.
`InsertSafetyTelegram(Int32)`	Creates the supplementary telegram with the parameterized telegram number and returns `true` if the telegram could be inserted. In the event of an error, an `EngineeringTargetInvocationException` is triggered.
`EraseTelegram(TelegramType)`	Returns `true` if the parameterized telegram could be deleted. Standard telegrams cannot be deleted. In the event of an error, an `EngineeringTargetInvocationException` is triggered.

Name	Description
`Find(TelegramType)`	Returns the Telegram (Page 1063) object if it could be found via the parameterized telegram type. `null` if the telegram is not found. **Example** `Telegram telegram = telegrams.Find(TelegramType.MainTelegram);`
`int IndexOf(TelegramType)`	Returns the index in the set of the elements for the queried instance.
`Parent`	Returns the reference to the higher-level class (`DriveObject`).
`bool Contains`	Determines as to whether the specified instance is included in the set of elements. `TRUE`: The container contains the instance `FALSE`: The container does not contain the instance
`IEnumerator GetEnumerator`	Facilitates `IEnumerator<DriveObject>` the iteration over the specified set of elements

Telegram

Telegram

The `Telegram` class allows access to the structure of a telegram from a drive object.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
`TelegramNumber`	`Int32`	read	-	Returns the number of the main telegram or specifies the number. A freely configurable telegram has the number 999.
`Type`	`enum`: TelegramType (Page 1065)	read	-	Returns the type of telegram as `Enum TelegramType`.
`Addresses`	AddressComposition (Page 1066)	read	-	Returns an `AdressComposition` with information on the address.

The following table describes the **methods** of the class:

Name	Description
`CanChangeTelegram(Int32)`	Returns `true` if the telegram can be changed to the parameterized standard type.
`GetSize(`AddressIoType (Page 1067)`)`	Returns the size of the inputs or outputs of the telegram.
`CanChangeSize(`AddressIoType (Page 1067)`, Int32, bool)`	Returns `true` if the size of the telegram can be changed as parameterized. Standard telegrams can only be enlarged. The retention of the previous telegram address is taken into account when the option is parameterized with `true`.
`ChangeSize(`AddressIoType (Page 1067)`, Int32, bool)`	Returns `true` if the size of the telegram could be changed as parameterized. The retention of the previous telegram address is taken into account when the option is parameterized with `true`.
`GetEnumerator`	Facilitates the iteration over the specified set of elements

Properties of Safety telegrams

The following table describes the additional **properties** of the "`Telegram`" class of the type "`SafetyTelegram`".

Name	Data type	Access mode	Access	Description
`Failsafe_FSourceAddress`	`UInt32`	read	Access only possible via `GetAttribute` or `SetAttribute`	PROFIsafe source address
`Failsafe_FDestinationAddress`	`UInt32`	read/ write	Access only possible via `GetAttribute` or `SetAttribute`	PROFIsafe destination address
`Failsafe_FIODBNumber`	`UInt32`	read/ write	Access only possible via `GetAttribute` or `SetAttribute`	Safety DB number, write access only possible if the property "`Failsafe_ManualAssignmentFIODBNumber`" is assigned the value "1"
`Failsafe_FIODBName`	`string`	read	Access only possible via `GetAttribute` or `SetAttribute`	Name Safety DB
`Failsafe_ManualAssignmentFIODBNumber`	`bool`	read/ write	Access only possible via `GetAttribute` or `SetAttribute`	Setting for manual assignment of the property "`Failsafe_FIODBNumber`"

Name	Data type	Access mode	Access	Description
Failsafe_FMonitoringtime	UInt32	read/ write	Access only possible via GetAttribute or SetAttribute	Safety monitoring time, write access only possible if the property "Failsafe_ManualAssignmentFMonitoringtime is set to "true"
Failsafe_ManualAssignmentFMonitoringtime	bool	read/ write	Access only possible via GetAttribute or SetAttribute	Setting for manual assignment of the property "Failsafe_FMonitoringtime"

The additional properties are accessed via GetAttribute and SetAttribute, e.g. GetAttribute("Failsafe_FSourceAddress").Uint32 is expected as return value.

TelegramType

TelegramType

The Enum TelegramType contains predefined telegram types.

Namespace: Siemens.Engineering.MC.Drives
Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **enum entries**:

Name	Description
MainTelegram	Main telegram: e.g. telegram 136
SupplementaryTelegram	Supplementary telegram: e.g. telegram 701
AdditionalTelegram	Additional telegram: free telegram
SafetyTelegram	Safety telegram: e.g. telegram 903

Note

The torque telegram (TorqueTelegram) is not supported, although it is available in the TIA Portal Openness API in the SINUMERIK context.

AddressComposition

AddressComposition

The `AddressComposition` class represents the address of a telegram.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
`IoType`	`enum`: AddressIoType (Page 1067)	read	-	Returns information on the type of the address.
`Context`	`enum`: AddressContext (Page 1067)	read	Access only possible via `GetAttribute` or `SetAttribute`	Returns information on the context of the address.
`StartAddress`	`Int32`	read/ write	-	Returns the start address of the telegram or defines the start address.
`Length`	`Int32`	read	-	Returns the length of the telegram.
`IndexOf`	`int32`	read	-	Returns the index in the set of the elements for the queried instance.
`Contains`	`bool`	read	-	Determines as to whether the specified instance is included in the set of elements. `TRUE`: The container contains the instance `FALSE`: The container does not contain the instance
`GetEnumerator`	`IEnumerator<DriveObject>`	read	-	Facilitates the iteration over the specified set of elements

Note

If you want to navigate back to the telegram via Address.Parent, instead of namespace `Siemens.Engineering.MC.Drives.Telegram`, namespace `Siemens.Engineering.MC.DriveConfiguration.Telegram` is called.

You can find further information about TIA Portal Openness libraries under "Standard libraries".

AddressContext

AddressContext

The Enum AddressContext contains information on the context of the address.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **enum entries**:

Name	Description
`AddressContext.None`	No context has been found for the address
`AddressContext.Device`	Context is a device address
`AddressContext.Head`	Context is a head address

You can find further information about TIA Portal Openness libraries under "Standard libraries".

AddressIoType

AddressIoType

The Enum AddressIoType contains information on the type of the address.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **enum entries**:

Name	Description
`AddressIoType.None`	The IO type cannot be used
`AddressIoType.Input`	Type is an input address
`AddressIoType.Output`	Type is an output address
`AddressIoType.Diagnosis`	Type is a diagnosis address
`AddressIoType.Substitute`	Type is a substitute address

You can find further information about TIA Portal Openness libraries under "Standard libraries".

5.21.5.3 ArchiveProvider

ArchiveProvider

The ArchiveProvider class is used to generate the PLC archives.

You can also generate archives via Openness if the Safety Integrated mode is activated.

Namespace: `Siemens.Engineering.HW.Utilities`
Assembly: `Siemens.Engineering.HW.Utilities` in `Siemens.Engineering.dll`

The following table describes the methods of the ArchiveProvider class:

Name	Parameter	Description

`Archive`	`DeviceItem plc`
	`FileInfo path`
	`Siemens.Engineering.MC.Sinumerik.SinumerikArchivationMode`
	`String comment` (optional)
	`String author` (optional)
		Creates a PLC archive

`Retrieve`	`FileInfo archivePath`	Consistently loading the archive into the TIA Portal

Note

Consistently loading from the F-CPU

Before generating an archive, ensure that you set option "Consistently loading from the F-CPU" in the Safety Administration editor under "Settings > Extended settings" so that you can also load devices with activated Safety Integrated mode as new station into the TIA Portal project.

(Function "Tools > Upload device from SINUMERIK archive > Upload device as new station (hardware and software)")

5.21.6 Code example

5.21.6.1 General

The following code examples describe the basic procedure for various applications. The code is not necessarily complete or compilable.

5.21.6.2 Executing the first steps in SINUMERIK

- Connect the TIA Portal Openness application with the TIA Portal.
- Open your project.

The following example shows how you can determine which SINUMERIK Toolbox version is installed.

Determining the SINUMERIK Toolbox version

```
using Siemens.Engineering;
if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINUMERK Toolbox") &&
sw.Version.Equals("V17")))
 {
  Console.WriteLine("SINUMERIK Toolbox is available");
 }
  // "V17" is the current SINUMERIK version started at October 2021.
```

5.21.6.3 Creating an NCU

You create a SINUMERIK NCU using the `CreateWithItem()` method of the `Devices Collection`. SINUMERIK NCUs are specified using method parameters. The format of the parameters is described in the following.

Creating an NCU

Parameter format for SINUMERIK NCU:

```
CreateWithItem(@"OrderNumber:mlfb/FirmwareVersion/",
"StandardSubrackName", "NameOfTheDevice")
```

The following example shows how you can create a SINUMERIK ONE NCU 1750 module.

Creating a SINUMERIK ONE NCU 1750 module

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open("..."); //The path of the project

Device NCUDevice = tiaproject.Devices.CreateWithItem("@OrderNumber:6FC5 317-5AA00-0AA0/
V6.15/", "NCU 1750", "TestDevice");
```

5.21.6.4 Creating an NX module

You create an NX module using the `Device CreateWithItem` method. You then connect the NX module to an NCU.

The following type identifiers are used for SINUMERIK NX modules:

SINUMERIK NX module	Type identifier
SINUMERIK NX10.3	OrderNumber:6SL3 040-1NC00-0Axx/Vy.z
SINUMERIK NX15.3	OrderNumber:6SL3 040-1NB00-0Axx/Vy.z

The following example shows how you can create a SINUMERIK NX module.

Creating an NX module

```
project.Devices.CreateWithItem("OrderNumber:6SL3040-1NC00-0AA0/
V5.2", "MyNXDevice", "TestDevice");
```

Version compatibility

The firmware version of the NX module must match the firmware version of SINAMICS Integrated - and must be compatible with the firmware version of the NCU.

An NX firmware version that deviates from SINAMICS Integrated cannot be assigned via Openness.

The following table lists the version compatibility for SINUMERIK ONE:

NCU firmware (SINUMERIK ONE)	SINAMICS Integrated/NX firmware
V6.15	V5.2
V6.20	
V6.21	
V6.22	

5.21.6.5 Connecting an NX module with NCU

Connecting an NX module with an NCU

To connect an NX module with an NCU via subnet type "`ProfibusIntegrated`", the "`NetworkInterface`" service must be loaded via the header module of NX.

The following example shows how you can load the "`NetworkInterface`" service.

Loading the NetworkInterface service

```
foreach (Device device in project.Devices)
{
  foreach (DeviceItem deviceItem in device.DeviceItems)
  {
   if (deviceItem.Classification == DeviceItemClassifications.HM)
   {
    var networkInterface = deviceItem.GetService<NetworkInterface>();
    // do something
   }
  }
}
```

The following example shows how you can connect an NX module with an NCU via "`ProfibusIntegrated`":

Connecting an NX module via ProfibusIntegrated

```
Subnet pbiSubnet = ...;
Node node = networkInterface.Nodes.FirstOrDefault();
node.ConnectToSubnet(pbiSubnet);
```

The DP address is assigned to the fixed NX in Openness via DRIVE-CLiQ port. Each port label has a fixed DP-integrated address.

The DP address must be assigned to the NX before connection to the NCU.

5.21.6.6 Accessing NCK events

NCK events

According to the SINUMERIK object model (Page 1049), the NCK module is a `DeviceItem`.

With the following attributes of the NCK module, you can access the NCK events and configure the NCK events in TIA Portal Openness:

Name	Data type	Access mode	Description
`HardwareInterruptNckToPlcSignalExchangeActive`	`Bool`	`r/w`	Activating/deactivating the event
`HardwareInterruptNckToPlcSignalExchangeEventName`	`String`	`r/w`	Event name
`HardwareInterruptNckToPlcSignalExchangeInterrupt`	`Siemens.Engineering.SW.Blocks.OB`	`r/w`	The OB assigned to the event
`HardwareInterruptNckToPlcSignalExchangePriority`	`Int32`	`r/w`	Event priority

All NCK events are triggered via `HardwareInterrupt` (hardware interrupt OB). The hardware interrupt OBs interrupt the cyclical program processing due to a hardware event.

The following example shows how to determine the event name for an NCK module:

Determining the event name for an NCK module

```
DeviceItem nck = ...;
string eventName =
(string)nck.GetAttribute("HardwareInterruptNckToPlcSignalExchangeEventName");
```

The following example shows how to set the hardware interrupt:

Setting the hardware interrupt

```
...  DeviceItem nck = ...;
OB ob40 = ...  try
{
  nck.SetAttribute("HardwareInterruptNckToPlcSignalExchangeInterrupt", ob40);
}
catch( (EngineeringException ex) )
{
// Handle setting failure
}
```

5.21.6.7 Creating archives

Creating archives for series commissioning

Using TIA Portal Openness, create and export SINUMERIK archives to simplify series commissioning, for example.

You create SINUMERIK archives via the TIA Portal project property `HwUtilities` with the `SinumerikArchiveProvider` **service.**

The following example shows how to call the `SinumerikArchiveProvider` service:

Calling SinumerikArchiveProvider

```
Project project = ...;
SinumerikArchiveProvider archiveProvider =
project.HwUtilities.Find("SinumerikArchiveProvider") as SinumerikArchiveProvider;

if (archiveProvider != null)
{
// Work with the provider
}
```

The following example shows how to create a PLC archive which contains the hardware information and all of the data blocks:

Creating a PLC archive

```
...
Siemens.Engineering.HW.DeviceItem plc = ...;

try {
    // The file extension is required
   string archivePath = string.Format(@"D:\some_path\{0}.dsf", plc.Name);

   // Comment and author arguments are optional
   archiveProvider.Archive(plc, new FileInfo(archivePath),
SinumerikArchivationMode.HardwareAndAllProgramBlocks);
}
catch (EngineeringTargetInvocationException ex)
{
// Handle archive failure
}
```

You can use the method as follows:

```
void Archive(DeviceItem plc, FileInfo path,
SinumerikArchivationMode content, String comment, String author)
```

The following example shows how to update the software portions of a previously created archive:

Updating portions of archives

```
...
Siemens.Engineering.HW.DeviceItem plc_1 = ...;
Siemens.Engineering.HW.DeviceItem plc_1_copy = ...;
try {
    // The file extension is required
    string archivePath = @"D:\some_path\SinumerikArchive.dsf";

    // Create a Sinumerik archive with HardwareAndAllProgramBlocks
    archiveProvider.Export(plc_1, new FileInfo(archivePath),
SinumerikArchivationMode.HardwareAndAllProgramBlocks);

   // Update the software part in the previously created archive using
UpdateProgramBlocksOfArchive method
archiveProvider.UpdateProgramBlocksOfArchive(plc_1_copy, new FileInfo(archivePath));
}
catch (EngineeringException ex)
{
   // Handle export failure
}
```

The following example shows how you can create SINUMERIK archives using the F-address assignment:

Creating a SINUMERIK archive using F-address assignment

```
...
Siemens.Engineering.HW.DeviceItemplc_1=...;
Siemens.Engineering.HW.DeviceItemplc_1_copy=...;

try{
    // The file extension is required
    stringarchivePath=@"D:\some_path\SinumerikArchive.dsf";

    // Create a SINUMERIK archive with F-address assigmnemt using
CreateFAddressAssignmentArchivemethod

archiveProvider.CreateFAddressAssignmentArchive(plc_1,plc_1_copy,newFileIn
fo(archivePath)[,"Comment","Author name"]);
}
catch(EngineeringTargetInvocationExceptionex){
// Handle archive failure
}
```

The following example shows how you can consistently load an archive into the TIA Portal (only the PLC component of SINUMERIK-DSF archives, type Setup):

Consistently loading archives into the TIA Portal

```
...
try{
    // The path of the
archivestringarchivePath=@"D:\some_path\SinumerikArchive.dsf";

   // Retrieve an already archived SINUMERIK device

DeviceretrievedDevice=archiveProvider.Retrieve(newFileInfo(archivePath));
}
catch(EngineeringTargetInvocationException ex)
{
  // Handle retrieve failure
}
```

5.21.6.8 Activating Safety Integrated

Activating Safety Integrated

With TIA Portal Openness, you can activate Safety Integrated (F-PLC) in the properties of the NCU.

Note

Effects on telegram configuration

The Safety Integrated mode used affects the telegram configuration because telegrams other than the ones used for inactive Safety Integrated mode are used in Safety Integrated plus (F-PLC) mode.

However, added or changed telegrams are retained if they are compatible with the newly selected Safety Integrated mode.

If applicable, after the mode change in the telegram configuration, ensure that any adjustments are still available.

You activate and deactivate Safety Integrated (F-PLC) with the `SafetyModeProvider` **service.**

Note

The PLC must be in offline mode if Safety Integrated (F-PLC) is being activated or deactivated.

Note

SafetyModeProvider is included in Namespace Siemens.Engineering.MC.Sinumerik .

The following example shows how to call the `SafetyModeProvider` service:

Calling SafetyModeProvider

```
...
Siemens.Engineering.HW.Device cnc = ...;
try
{
   SafetyModeProvider provider = cnc.GetService<SafetyModeProvider>();
   //Perform the safety mode change:
   provider.SetSafetyMode(SafetyMode.DbSI);
}
catch( (EngineeringException ex) )
  {
  // Handle safety mode change failure
   }
```

The following example shows how to call the current Safety Integrated setting on the device:

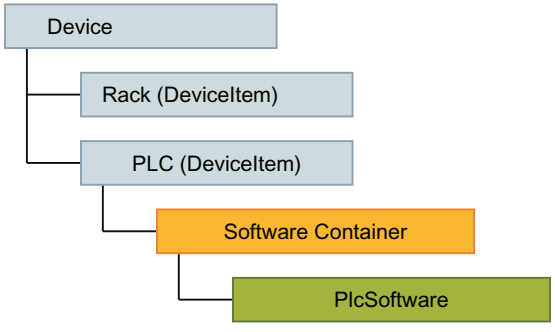
Calling the safety setting of the device

```
...
Siemens.Engineering.HW.Device cnc = ...;
try
{
   SafetyModeProvider provider = cnc.GetService<SafetyModeProvider>();
   //Query the safety mode:
   SafetyMode safetyMode = provider.CurrentMode;
}
catch( (EngineeringException ex) )
{
// Handle any failure
  }
```

5.21.6.9 Accessing PLC software

Overview

The following diagram shows the software container and the PLC software in the object model (Page 1049):



The following code example shows how to find a PLC, irrespective of its specific implementation (integrated SINUMERIK PLC, SIMATIC PLC, software PLC in the PC) based on the "CPU" property:

Finding PLCs

```
Device ncuDevice = ...
DeviceItem plc = GetPlc(ncuDevice.DeviceItems);

   ...
DeviceItem GetPlc(DeviceItemComposition deviceItems)
{
   if (deviceItems.Count == 0)
   {
       return null;
   }
   foreach (var deviceItem in deviceItems)
   {
       if (deviceItem.Classification == DeviceItemClassifications.CPU)
           return deviceItem;

       DeviceItem plc = GetPlc(deviceItem.DeviceItems);
       if (plc != null)
           return plc;
   }

   return null;
}
```

Find out more about accessing the PLC software container under "Accessing software goals".

5.21.6.10 Importing SINUMERIK PLC alarm texts

The SINUMERIKAlarmTextProvider class is used to import the texts of the SINUMERIK-DB2 alarms in TS or CSV format.

Note

SinumerikAlarmTextProvider is included in Namespace Siemens.Engineering.MC.Sinumerik .

You import SINUMERIK PLC alarm texts via Navigator `AlarmTextImporter` with the `SinumerikAlarmTextProvider` service.

The following example shows how you can import the texts of the SINUMERIK DB2 alarms from a SINUMERIK NCU via `SinumerikAlarmTextProvider`.

Importing texts via SinumerikAlarmTextProvider

```
...Siemens.Engineering.HW.Devicencu=...;
System.Collections.IEnumerable<System.IO.FileInfo>inputFiles=...;
try
{
  // Get SinumerikAlarmTextProvider service from a SINUMERIK device
  SinumerikAlarmTextProviderprovider=ncu.GetService<SinumerikAlarmTextProvider>();
  // Call import function through the navigator:
  provider.AlarmTextImporter.ImportAlarmTexts(inputFiles);
}
catch(EngineeringExceptionex)
{
// Handle import failures
}
```

Information on importing SINUMERIK User Alarms (in user text lists) can be found in the TIA Portal help under Exporting/importing PLC message text lists.

5.21.6.11 Examples for Namespace Siemens.Engineering.MC.DriveConfiguration

Preparing telegrams

The drive communication of a SINUMERIK NCU is realized using telegrams via the SINAMICS Integrated subcomponent and, if applicable, via additionally connected NX modules.

Note

The SINUMERIK NCU and a SINAMICS Integrated are located at the same level in the TIA Portal Openness object model, and appear under "`DeviceComposition`" as two different devices.

Use "`DriveObjectCollection`" to configure a telegram. "`DriveObjectCollection`" is a service of the drive object (device element) of the actual header module (device element).

To start the "`DriveObjectCollection`" service, navigate to SINAMICS Integrated or to the header module of the NX module. The hierarchy between devices and device elements is identical for SINAMICS Integrated and NX modules.

The following example shows how to find "DriveObjectCollection" via the "Header module" property:

Finding DriveObjectCollection via header module

```
foreach (Device device in project.Devices)
{
   foreach (DeviceItem deviceItem in device.DeviceItems)
   {
     if (deviceItem.Classification == DeviceItemClassifications.HM)
     {
       var driveObjectCollection = deviceItem.GetService<DriveObjectCollection>();
             // do something
      }
    }
}
```

The SINUMERIK NCU contains a SINAMICS Integrated proxy object with references to SINAMICS Integrated.

To access a SINAMICS Integrated device or an NX module, from the SINUMERIK NCU navigate to the DP Integrated interface via NCK. Then determine the PROFIBUS master system and navigate to the connected slave.

Inserting telegrams

The following example indicates how you can insert a telegram. You require a drive object for access.

You distinguish between the telegram types using their IDs.

Inserting a telegram

```
using Siemens.Engineering.MC.DriveConfiguration;
TelegramComposition telegrams = drvObj.Telegrams;

//Create telegram
drvObj.Telegrams.Create(TelegramId.Supplementary_Telegram_701);

}
```

Working with Safety telegrams

The following example shows how to insert a Safety telegram. You require a drive object for access.

Inserting a Safety telegram and accessing telegram attributes

```
using Siemens.Engineering.MC.DriveConfiguration;
```

Inserting a Safety telegram and accessing telegram attributes

```
TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
drvObj.Telegrams.Create(TelegramId.PROFIsafe_Standard_Telegram_30);

// Get and set safety telegram attributes
uint Failsafe_FDestinationAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FDestinationAddress");
uint Failsafe_FSourceAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FSourceAddress");
uint Failsafe_FIODBNumber = (uint)safetyTelegram.GetAttribute("Failsafe_FIODBNumber");
string Failsafe_FIODBName = safetyTelegram.GetAttribute("Failsafe_FIODBName").ToString();
uint Failsafe_FMonitoringtime =
(uint)safetyTelegram.GetAttribute("Failsafe_FMonitoringtime");
uint Failsafe_ManualAssignmentFIODBNumber =
(uint)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFIODBNumber");
bool Failsafe_ManualAssignmentFMonitoringtime =
(bool)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFMonitoringtime");

// Set safety telegram attributes
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFIODBNumber", 1);
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFMonitoringtime", true);
safetyTelegram.SetAttribute("Failsafe_FIODBNumber", 40000);
safetyTelegram.SetAttribute("Failsafe_FMonitoringtime", 200);
safetyTelegram.SetAttribute("Failsafe_FDestinationAddress", 15);

const int newSafetyTelegramId= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramId)) {
safetyTgrm.TelegramId = newSafetyTelegramId; }
```

The following example shows how to delete a safety telegram.

Deleting a safety telegram

```
using Siemens.Engineering.MC.DriveConfiguration;
//Remove Safety telegram
 drvObj.Telegrams.DeleteTelegram(TelegramType.SafetyTelegram);
```

Extending telegrams

The following example shows how to insert an extension and change the size of a standard telegram. You require a drive object for access.

Inserting an extension and changing the size of a standard telegram

```
using Siemens.Engineering.MC.DriveConfiguration;
```

Inserting an extension and changing the size of a standard telegram

```
TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelegramId);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: "
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
  if (address.IoType == AddressIoType.Output)
  {
    Console.WriteLine("The Setpoint channel-specific IO start address of
the telegram on the connected PLC is: " + address.StartAddress);
  }
  else if(address.IoType == AddressIoType.Input)
  {
    Console.WriteLine("The Actual value channel-specific IO start address
of the telegram on the connected PLC is: " + address.StartAddress);
  }
}

// Create an additional telegram
if (drvObj.Telegrams.CreateAdditionalTelegram(2,4))
{
  drvObj.Telegrams.CreateAdditionalTelegram(2,4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
  mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}
```

5.21.6.12 Examples for Namespace Siemens.Engineering.MC.Drives

Preparing telegrams

The drive communication of a SINUMERIK NCU is realized using telegrams via the SINAMICS Integrated subcomponent and, if applicable, via additionally connected NX modules.

Note

The SINUMERIK NCU and a SINAMICS Integrated are located at the same level in the TIA Portal Openness object model, and appear under "`DeviceComposition`" as two different devices.

Use "`DriveObjectContainer`" to configure a telegram. "`DriveObjectContainer`" is a service of the drive object (device element) of the actual header module (device element).

To start the "DriveObjectContainer" service, navigate to SINAMICS Integrated or to the header module of the NX module. The hierarchy between devices and device elements is identical for SINAMICS Integrated and NX modules.

The following example shows how to find "DriveObjectContainer" via the "Header module" property:

Finding DriveObjectContainer via header module

```
foreach (Device device in project.Devices)
{
   foreach (DeviceItem deviceItem in device.DeviceItems)
   {
     if (deviceItem.Classification == DeviceItemClassifications.HM)
     {
       var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
             // do something
      }
    }
}
```

The SINUMERIK NCU contains a SINAMICS Integrated proxy object with references to SINAMICS Integrated.

To access a SINAMICS Integrated device or an NX module, from the SINUMERIK NCU navigate to the DP Integrated interface via NCK. Then determine the PROFIBUS master system and navigate to the connected slave.

Inserting and removing telegrams

The following example indicates how you can insert a telegram. You require a drive object for access.

Inserting telegrams and accessing telegram attributes

```
using Siemens.Engineering.MC.Drives;
```

Inserting telegrams and accessing telegram attributes

```
TelegramComposition telegrams = drvObj.Telegrams;

//Add telegram
drvObj.Telegrams.InsertAdditionalTelegram(tgrmNumber);

//Find telegram
Telegram telegram = drvObj.Telegrams.Find(TelegramType.MainTelegram);

/Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram;tgrmNumber)(

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
 uint watchDogTime =
(uint)safetyTgrm.GetAttribute("Failsafe_FMonitoringtime");

safetyTgrm.SetAttribute("Failsafe_FMonitoringtime", 300);

const int newSafetyTelegramNumber= 902;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber)) {
safetyTgrm.TelegramNumber = newSafetyTelegramNumber; }

//Add supplementary telegram
const int tgrmNumber = 701;
drvObj.Telegrams.InsertSupplementaryTelegram(tgrmNumber);

Telegram telegram =
drvObj.Telegrams.Find(TelegramType.SupplementaryTelegram);
```

The following example indicates how you can remove a telegram.

Remove telegram

```
using Siemens.Engineering.MC.Drives;
//Remove safety telegram
 drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);

//Remove supplementary telegram
 drvObj.Telegrams.EraseTelegram(TelegramType.SupplementaryTelegram);
```

Note

You can change, but not delete, a main telegram (`MainTelegram`).

Working with Safety telegrams

The following example shows how to insert a Safety telegram. You require a drive object for access.

Inserting a Safety telegram and accessing telegram attributes

```
using Siemens.Engineering.MC.Drives;
TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram(tgrmNumber);

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
uint Failsafe_FDestinationAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FDestinationAddress");
uint Failsafe_FSourceAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FSourceAddress");
uint Failsafe_FIODBNumber = (uint)safetyTelegram.GetAttribute("Failsafe_FIODBNumber");
string Failsafe_FIODBName = safetyTelegram.GetAttribute("Failsafe_FIODBName").ToString();
uint Failsafe_FMonitoringtime =
(uint)safetyTelegram.GetAttribute("Failsafe_FMonitoringtime");
uint Failsafe_ManualAssignmentFIODBNumber =
(uint)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFIODBNumber");
bool Failsafe_ManualAssignmentFMonitoringtime =
(bool)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFMonitoringtime");

// Set safety telegram attributes
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFIODBNumber", 1);
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFMonitoringtime", true);
safetyTelegram.SetAttribute("Failsafe_FIODBNumber", 40000);
safetyTelegram.SetAttribute("Failsafe_FMonitoringtime", 200);
safetyTelegram.SetAttribute("Failsafe_FDestinationAddress", 15);

const int newSafetyTelegramNumber= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber)) {
safetyTgrm.TelegramNumber = newSafetyTelegramNumber; }
```

The following example shows how to delete a safety telegram.

Deleting a safety telegram

```
using Siemens.Engineering.MC.Drives;
//Remove Safety telegram
 drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);
```

Extending telegrams

The following example shows how to insert an extension and change the size of a standard telegram. You require a drive object for access.

Inserting an extension and changing the size of a standard telegram

```
using Siemens.Engineering.MC.Drives;
```

Inserting an extension and changing the size of a standard telegram

```
TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelegramNumber);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: "
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
  if (address.IoType == AddressIoType.Output)
  {
    Console.WriteLine("The Setpoint channel-specific IO start address of
the telegram on the connected PLC is: " + address.StartAddress);
  }
  else if(address.IoType == AddressIoType.Input)
  {
    Console.WriteLine("The Actual value channel-specific IO start address
of the telegram on the connected PLC is: " + address.StartAddress);
  }
}

// Add an additional telegram
if (drvObj.Telegrams.CanInsertAdditionalTelegram(2,4))
{
  drvObj.Telegrams.InsertAdditionalTelegram(2,4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
  mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}
```

5.22 Functions for SINUMERIK MC

5.22.1 Introduction

For SINUMERIK MC, only the navigation to the PLC software container is available in Openness.

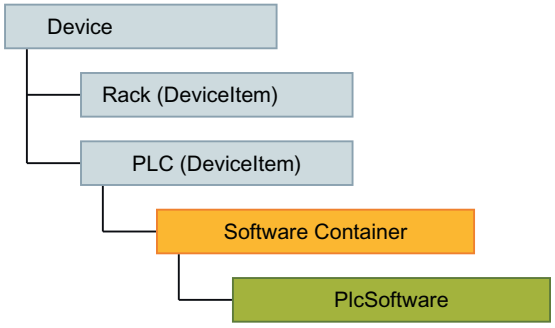
Other Openness functions are neither available for control-specific components nor for PC-specific components of a SINUMERIK MC.

5.22.2 Code example

5.22.2.1 Accessing PLC software

Overview

The following diagram shows the software container and the PLC software in the object model:



The following code example shows how to find a PLC, irrespective of its specific implementation (integrated SINUMERIK PLC, SIMATIC PLC, software PLC in the PC) based on the "CPU" property:

Finding PLCs

```
Device ncuDevice = ...
DeviceItem plc = GetPlc(mcuDevice.DeviceItems);

   ...
DeviceItem GetPlc(DeviceItemComposition deviceItems)
{
   if (deviceItems.Count == 0)
   {
       return null;
   }
   foreach (var deviceItem in deviceItems)
   {
       if (deviceItem.Classification == DeviceItemClassifications.CPU)
           return deviceItem;

       DeviceItem plc = GetPlc(deviceItem.DeviceItems);
       if (plc != null)
           return plc;
   }

   return null;
}
```

Find out more about accessing the PLC software container under "Accessing software goals".

5.23 Functions for Startdrive

5.23.1 Introduction

Using TIA Portal-Openness, you automate the engineering and control the TIA Portal using a program that you created yourself.

In this help document, you can find a lot of information and code examples for this program that you generate yourself. You can also generate and use your own programs for the TIA Portal "Startdrive" application.

Before you configure your own program for Startdrive from the subsequently listed code example, please carefully observe the general information relating to Openness, which you can find under the following keywords in this information system:

- Preconditions for using TIA Portal Openness
- Install TIA Portal Openness
- Access the TIA Portal
- TIA Portal Openness object model
- Programming steps

Note

Activated project protection (UMAC)

If you want to edit a project for which a project protection is activated via "Openness API", you need to have the corresponding access rights for this purpose. Your user account must have the following function rights:

- Change project via Openness API
- Open and edit the project
- Open the project read-only

Additional function rights are required for technology objects. If you do not have any access rights, contact your administrator.

For drives as of SINAMICS FW V6.1 or higher, you require the same access rights for your user account to access the drive online as in Startdrive.

Additional information on access rights and on user administration in general is provided on Page "User administration and Security".

5.23.2 Security information

5.23.2.1 Encrypted communication with SecureString passwords

Note

SecureString passwords for secure communication

In order to secure communication between Openness API and the TIA Portal when using Startdrive Openness functions, use passwords which have been encrypted with a SecureString.

Note

Direction of communication

It is only possible to use SecureString passwords when downloading Openness functions to the TIA Portal and not in the opposite communication direction.

Note

A SecureString object should never be constructed from a String because the sensitive data is already subject to the memory persistence consequences of the immutable String class. The best way to construct a SecureString object is from a character-at-a-time unmanaged source, such as the Console.ReadKey method.

Note

The API user is responsible for security measures when handling passwords using code.

You can find more information on configuring SecureString passwords in the code examples download (Page 1115) and Creating SecureString passwords (Page 1143).

See also

Passwords in TIA Openness (Page 1087)

5.23.2.2 Passwords in TIA Openness

Handling passwords in TIA Openness

Password handling in Startdrive Openness is based on TIA Openness.

Please carefully observe the general information relating to passwords in TIA Openness, which you can find under the following keywords in this help:

- Defining the display password
- Handling PLC passwords linked to blocks
- Password protection for PLCs

- Setting protection levels
- Block protection and unprotection
- Loading hardware, software and files into the PLC device

See also

Encrypted communication with SecureString passwords (Page 1087)

5.23.3 TypeIdentifier - identifier of the components

Each version of any Startdrive component comprises a unique number, which is called the TypeIdentifier . In the Openness program code, you can use these TypeIdentifiers in order to uniquely identify and name a component.

In Startdrive, the display of the TypeIdentifier is optional, and deactivated as default.

Activating the display of the TypeIdentifier in Startdrive

1. In the Startdrive project view, select menu "Options > Settings".
   The "Settings" configuration area opens.
2. Select "Hardware configuration" in the secondary navigation.
3. Activate the option "Activate display of the Type Identifier for devices and modules".
   The display of the TypeIdentifier is now active.

Reading out the TypeIdentifier in Startdrive

In a Startdrive project, the TypeIdentifier can be read out at the following locations when the display is active:

- For all components (in the inspector window):
- For Control Units (when creating a drive device)

To read out the TypeIdentifier for a component in the inspector window, proceed as follows:

1. In the device view of the Startdrive project, double-click on the required component.
   The inspector window opens. The active component version is shown in the list.
2. The associated TypeIdentifiers are listed in the outer right-hand column.
   Example: `OrderNumber:6SL3131-7TE23-6Axx`
   Copy this TypeIdentifier to your Openness application.

Proceed as follows to read out the TypeIdentifier for a Control Unit:

1. In the Startdrive project navigation, double click on "Add new device".
   The dialog with the same name opens.
2. Select the required control module from the selection.
   The TypeIdentifier (under the article number and firmware number) for the corresponding Control Unit is displayed to the right in the detailed display.
   Example: `OrderNumber:6SL3040-1MA01-0Axx/V5.2/S120`
3. Copy this TypeIdentifier to your Openness application.

5.23.4 References

5.23.4.1 AddressComposition

AddressComposition

The `AddressComposition` class represents the address of a telegram.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public sealed class AddressComposition
```

The following table describes the **properties** of the class:

Name	Data type	Description
`IoType`	AddressIoType (Page 1090)	Returns information on the type of the address.
`Context`	AddressContext (Page 1090)	Returns information on the context of the address.
`StartAddress`	`Int32`	Returns the start address of the telegram or specifies it.
`Length`	`Int32`	Returns the length of the telegram.

See also

TelegramType (Page 1108)

5.23.4.2 AddressContext

AddressContext

The Enum AddressContext contains information on the context of the address.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public enum AddressContext
```

The following table describes the **enum entries**:

Name	Description
`AddressContext.None`	No context has been found for the address
`AddressContext.Device`	Context is a device address
`AddressContext.Head`	Context is a head address

5.23.4.3 AddressIoType

AddressIoType

The Enum AddressIoType contains information on the type of the address.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public enum AddressIoType
```

The following table describes the **enum entries**:

Name	Description
`AddressIoType.None`	The IO type cannot be used
`AddressIoType.Input`	Type is an input address
`AddressIoType.Output`	Type is an output address
`AddressIoType.Diagnosis`	Type is a diagnosis address
`AddressIoType.Substitute`	Type is a substitute address

5.23.4.4 Commissioning

Commissioning

The `Commissioning` class is used for commissioning a drive. It can be applied to `DriveFunctionInterface` or `OnlineDriveFunctionInterface`. If the user uses the function for a SINAMICS S drive, it returns the return value zero.

The following table describes the **methods** of the class:

Name	Parameter	Return value	Description
`SetSimoGearMlfb`	string simoGearMlfb	bool	Sets the SIMOGEAR Article No. of a drive.

See also

Setting the SIMOGEAR article number for G115D drives (Page 1146)

5.23.4.5 ConfigurationEntry

ConfigurationEntry

Class `ConfigurationEntry` is used to save parameter data, which can be determined from the `ConfigurationEntryCompositions` of a motor or encoder configuration.

The following table describes the **properties** of the class:

Name	Data type	Description
`EnumValueList`	`IDictionary<it, string>`	Returns a list with possible values of the enum parameter.
`MaxValue`	`object`	Maximum value of the `ConfigurationEntry`.
`MinValue`	`object`	Minimum value of the `ConfigurationEntry`.
`Name`	`string`	Name of the `ConfigurationEntry`.
`Number`	`int`	Numerical representation of the name for `ConfigurationEntry`.
`Description`	`string`	Description of the `ConfigurationEntry`.
`Parent`	`IEngineeringObject`	Engineering Object model-parent element of this object.
`Unit`	`string`	Unit of the `ConfigurationEntry`.
`Value`	`object`	Value of the `ConfigurationEntry`.

5.23.4.6 DriveDataEncryption (from SINAMICS FW V6.1)

Class `DriveDataEncryption` allows the encryption of drive data (DDE, Drive Data Encryption) for SINAMICS drives from firmware version V6.1 to be activated and deactivated with Openness.

The following table describes the **methods** of class `DriveDataEncryption`:

Name	Parameter	Return value	Description	Exceptions
`Activate`	SecureString password	bool	Activate `DriveDataEncryption` with password.	-
`Deactivate`	SecureString password	bool	Deactivate `DriveDataEncryption` with password.	-

Note

`Security` class

The `Security` class is a higher-level class. The class is used in the background to access all APIs that are involved in security.

5.23.4.7 DriveDomainFunctions

DriveDomainFunctions

Class `DriveDomainFunctions` is used to restore the factory settings or the backup of the RAM content to the ROM.

It can only be applied to an `OnlineDriveFunctionInterface` object.

For G120 drives, the `DriveDomainFunctions` object can only be accessed if the Power Module is connected to the device. Otherwise, null or an exception is returned.

The following table describes the **methods** of the class:

Name	Description
`PerformFactoryReset`	This method is responsible for restoring the factory settings.
`PerformRAMtoROMCopyAllDriveObject`	The date of all drive objects is written from the RAM to the memory card/ hard disk.

The following table describes the **properties** of the class:

Name	Data type	Description
`Parent`	`IEngineeringObject`	Engineering Object model-parent element of this object.

5.23.4.8 DriveObject

DriveObject

The `DriveObject` class allows access to the drive object. Access to the drive parameters or the telegram is possible via the drive object, for example.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public sealed class DriveObject
```

The following table describes the **properties** of the class:

Name	Data type	Description
`Parameters`	DriveParameter-Composition (Page 1096)	Returns a list with the available parameters of the drive object.
`Telegrams`	TelegramComposition (Page 1107)	Returns a list with the available telegrams of the drive object. The list can be changed with class `TelegramComposition`.

See also

Determining a drive object (Page 1113)

5.23.4.9 DriveObjectActivation

DriveObjectActivation

Class `DriveObjectActivation` is used to activate the modules or determine the module status. It can be applied to `DriveObjectFunctions` from `DriveFunctionInterface` or `OnlineDriveFunctionInterface`.

The following table describes the **methods** of the class:

Name	Description
`ChangeActivationState(DriveObjectActivationState)`	Changes the activation status of the drive object. Returns false if the operation cannot be completed. Possible status values: • Deactivate • Activate • DeactivateAndNotPresent

The following table describes the **properties** of the class:

Name	Data type	Description
`ActivationState`	`DriveObjectActivationState`	Returns the actual status value.
`IsActive`	`Boolean`	For an active drive object, returns true.

5.23.4.10 DriveObjectContainer

DriveObjectContainer

The `DriveObjectContainer` is a service of the drive object (`DeviceItem`) for the current device (`Device`).

The following table describes the **navigators** of the `DriveObjectContainer`:

Name	Data type	Description
`DriveObjects`	DriveObjectComposition (Page 1096)	Returns a list with the available drive objects. The drive objects allow access to the drive parameters and telegrams.
`Parent`	IEngineeringObject	Engineering Object model-parent element of this object.

5.23.4.11 ModuleAccessPoint

ModuleAccessPoint

The `ModuleAccessPoint` is a service of the drive object (`DeviceItem`) for the current device (`Device`).

The following table describes the **properties** of the Hardware Identifier of the `ModuleAccessPoint`:

Name	Data type	Access	Description
`Hardware identifier`	`IEnumerable<IBrowsable>`	ReadOnly	Returns the list of hardware identifiers.

5.23.4.12 DriveObjectTypeHandler

DriveObjectTypeHandler

Class `DriveObjectTypeHandler` is used to switch over the drive object type for every drive object and to determine the drive object type - as well as all possible drive object types of the actual drive object. It can only be applied to `DriveFunctionInterface`.

The following table describes the **methods** of the class:

Name	Description
`ChangeDriveObjectType((target)DriveObjectType)`	Changes the actual type of the drive object to a new type that can be selected. Returns false if the operation cannot be completed.

The following table describes the **properties** of the class:

Name	Data type	Description
PossibleDriveObjectTypes	DriveObjectTypeComposition	The use of targetDriveObjectType results in an exception on the actual drive object.
CurrentDriveObjectType	DriveObjectType	Indicates the currently assigned drive object type.

5.23.4.13 DriveParameter

DriveParameter

The DriveParameter class allows access to a drive parameter. Not all drive parameters are available for access via Openness.

Namespace: Siemens.Engineering.MC.Drives
Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public sealed class DriveParameter
```

The following table describes the **properties** of the class:

Name	Data type	Description
ArrayIndex	Int32	Returns the index of an array parameter. Value range: 0-7FFF The index is -1 for parameters without array **Example** p108[4].15 par.ArrayIndex produces 4
ArrayLength	Int32	Returns the number of array elements. The index is 0 for parameters without an array
Bits	DriveParameter-Composition (Page 1096)	Returns an DriveParameter object for one bit of the parameter. In this way, for example, the value or the name of the bit parameter can be read from a bit parameter. **Example** DriveParameter param133 = cu.Parameters.Find(133, 0); DriveParameter param133Bit1 = param133.Bits[1]; String paramName = param133Bit1.Name;
EnumValueList	IDictionary<int, string>	Returns a list with possible values of the enum parameter. For example <1, [1] Quick commissioning> null, if the parameter is not a parameter of the Enum type.
MaxValue	Object	Returns the maximum value for the currently selected unit.
MinValue	Object	Returns the minimum value for the currently selected unit.

Name	Data type	Description
`Name`	`string`	Returns the name of the parameter. E.g. `"p108[0].2"`
`ParameterText`	`string`	Returns the text of the short description for the parameter.
`Number`	`Int32`	Returns the number of the parameter. **Example** `p108[0].2` The return value is 108
`Unit`	`string`	Returns the unit of the parameter as text.
`Value`	`Object`	Returns the offline/online value of the parameter or writes a value onto the parameter. If the event of write errors, an `EngineeringTargetInvocationException` is triggered. **Examples** • `P2080Bit6.Value = 0;` • `P2080Bit6.Value = cu.Parameters.Find("r19");` **BICO source** The parameters of a BICO source can only be read **BICO signal sinks** Possible values are 0, 1 or a `DriveParameter` object. A `DriveParameter` object is returned when the BICO signal sink is connected to a different parameter. See also example Reading and writing BICO parameters (Page 1114).

See also

Reading and writing parameters (Page 1140)

5.23.4.14 DriveParameterComposition

DriveParameterComposition

The `DriveParameterComposition` class allows access to parameters of the drive. Not all drive parameters are approved for access via Openness.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public sealed class DriveParameterComposition
```

The following table describes the **methods** of the class:

Name	Description
`Find(string)`	Returns the DriveParameter (Page 1095) object for which a search is being made via the name. `null` if the parameter is not found **Example** `Find("P108[1]");`
`Find(UInt16, Int32)`	Returns the DriveParameter (Page 1095) object for which a search is being made via the parameter index and array index. `null` if the parameter is not found **Examples** • `cu.Find(108, 1);` • `cu.Find(51, -1);`

5.23.4.15 HardwareProjection

Class `HardwareProjection` is responsible for commissioning the motor and the encoder. The object can be found for `DriveFunctionInterface` and `OnlineDriveFunctionInterface`.

For G120 drives, motors and encoders can be configured both online and offline. On the other hand, for S120 drives, configuration is only possible offline.

For G120 drives, the `HardwareProjection` object can only be accessed if the Power Module is inserted in the drive device. Otherwise, when calling functions `HardwareProjection`, a null or an exception is returned.

For an offline configuration, use the hardware configuration of the `DriveFunctionInterface`. For an online configuration, use the hardware configuration of the `OnlineDriveFunctionInterface`.

Namespace: `Siemens.Engineering.MC.Drives`
`Siemens.Engineering.MC.Drives.DFI`
`Siemens.Engineering.MC.Drives.Enums`

Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **methods** of the class:

Name	Parameter	Return value	Description	Exceptions
`SetMotorType`	MotorType type, ushort driveData-Set	bool	Sets the motor type at the Control Unit (only for G120).	Only available for G120 drives.

`GetCurrentMotorConfiguration`
For S120:
`GetCurrentMotorConfiguration(ushort motorDataset)`
	ushort driveData-Set	MotorConfigura-tion	Depending on the data set number of the drive, determines the currently existing configuration area.	-

Name	Parameter	Return value	Description	Exceptions
`ProjectMotorConfiguration`	MotorConfiguration motConfig, ushort driveDataSet	bool	Configures the motor configuration of a drive device depending on the data set number of the drive.	-
`SetEncoder`	EncoderType type, EncoderInterface interfaceType, AbsoluteIncrementalFlag absIncFlag, RotaryLinearFlag rotLinFlag, ushort encoderNumber	bool	Sets the encoder at the Control Unit (only for G120).	Only available for G120 drives.
`GetCurrentEncoderConfiguration`	ushort encoderNumber	EncoderConfiguration	Depending on the data set number of the encoder, determines the currently existing configuration area.	-
`ProjectEncoderConfiguration`	EncoderConfiguration encConfig, ushort enccoderNumber	bool	Configures the motor configuration of a drive device depending on the data set number of the encoder.	-

The following table describes the **properties** of the class:

Name	Data type	Access	Description
`Parent`	`IEngineeringObject`	ReadOnly	Engineering Object model-parent element of this object.

See also

Connecting technology objects with hardware modules via telegram objects (Page 1149)

5.23.4.16 EncoderConfiguration

EncoderConfiguration

Class `EncoderConfiguration` saves data of non-Siemens encoders.

- The user must populate the `ConfigurationEntryComposition` object.
- Object `RequiredConfigurationEntries` must also be populated.

Namespace: `Siemens.Engineering.MC.Drives`
`Siemens.Engineering.MC.Drives.DFI`
`Siemens.Engineering.MC.Drives.Enum`

Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Access	Description
RequiredConfigurationEntries	ConfigurationEntryComposition	ReadOnly	Accessible configuration inputs of Encoder-Configuration.
Parent	IEngineeringObject	ReadOnly	Engineering Object model-parent element of this object.
RequiredConfigurationEntries	IEnumerable<IBrowsable>	Writeable	Returns all encoder parameters that can be used, with the exception of p404.0 and/or p404.1.
EncoderTypes	IEnumerable<IBrowsable>	ReadOnly	Depending on the encoder, returns parameter p404.0 and/or p404.1.

The following table describes the **methods** of the class:

Name	Parameter	Return value	Description
SetEncoderType	RotaryLinearFlag enum	bool	Defines the non-DRIVE-CLiQ encoder type to be either rotary or linear .
SetEncoderType	RotaryLinearFlag enum, AbsoluteIncrementalFlag enum	bool	Defines the DRIVE-CLiQ encoder type to be either rotary or linear as well as absolute or incremental. Not intended for use with non-DRIVE-CLiQ encoders. G220 drives with firmware version V6.2 only support absolute or incremental encoder types. This API can be used for setting.

5.23.4.17 MotorConfiguration

MotorConfiguration

Class `MotorConfiguration` is responsible for commissioning motors and encoders.

Namespace: `Siemens.Engineering.MC.Drives`
`Siemens.Engineering.MC.Drives.DFI`
`Siemens.Engineering.MC.Drives.Enums`

Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

Response for Siemens motors:

Action `SetMotortype` must be called to set the motor type. This action comprises the Enum `MotorType` and a number, which represents the `DriveDatasetNumber`.

The action `SetMotortype` is not available for S120 drives. For these drives, the motor type is set using the hardware catalog by applying the `PlugNew()` method.

The following table describes the enums:

Enum name	Description
`NoMotor`	0: No motor
`InductionMotor`	1: Induction motor
`SynchronousMotor`	2: Synchronous motor
`NoCodeNumber1LE1InductionMotor`	10: 1LE1 induction motor (not a code number)
`NoCodeNumber1LG6InductionMotor`	13: 1LG6 induction motor (not a code number)
`NoCodeNumber1xx1SIMOTICSFDInductionMotor`	14: 1xx1 SIMOTICS FD induction motor (not a code number)
`NoCodeNumber1LA7InductionMotorNoCodeNumber`	17: 1LA7 induction motor (not a code number)
`MotorSeriesNumber1LA81PQ8StandardInduction`	18: 1LA8 / 1PQ8 standard induction motor series
`NoCodeNumber1LA9InductionMotor`	19: 1LA9 induction motor (not a code number)

Response for non-Siemens motors:

Class `MotorConfiguration` is used to save data of non-Siemens motors. It contains 2 objects `ConfigurationEntryComposition`, which, when commissioning, must be populated with the corresponding motor data. Object `RequiredConfigurationEntries` must also be populated. Object `OptionalConfigurationEntries` does not have to be populated.

The following table describes the **properties** of the class:

Name	Data type	Description
`RequiredConfigurationEntries`	`ConfigurationEntryComposition`	Accessible configuration entries of this motor configuration.
`OptionalConfigurationEntries`	`ConfigurationEntryComposition`	Accessible configuration entries of this motor configuration.
`Parent`	`IEngineeringObject`	Engineering object model-parent elements of this object (only S120 drives).

5.23.4.18 OnlineDriveObject

OnlineDriveObject

The `OnlineDriveObject` class allows online access to the drive object. Drive parameters can be accessed via the drive object.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public sealed class OnlineDriveObject
```

The following table describes the **properties** of the class:

Name	Data type	Description
`Parameters`	DriveParameter-Composition (Page 1096)	Returns a list with the available parameters of the online drive object. `null`, if the mode is "Offline". In offline mode, an exception is triggered when a method is called or for a write access to a parameter.

See also

Determining a drive object (Page 1113)

Reading and writing parameters (Page 1140)

Reading and writing parameters online (Page 1142)

5.23.4.19 OnlineDriveObjectContainer

OnlineDriveObjectContainer

The `OnlineDriveObjectContainer` is a service of the drive object (`DeviceItem`) for the current device (`Device`).

The following table describes the **navigators** of the `OnlineDriveObjectContainer`:

Name	Data type	Description
`OnlineDriveObjects`	`OnlineDriveObjectComposition`	Returns a list with the available online drive objects (OnlineDriveObject (Page 1100)). The drive objects allow access to the drive parameters.
`DriveDataEncryption`	`Siemens.Engineering.MC.Drives.SecurityObjects.DriveDataEncryption`	Configures the encryption of drive data (DDE, drive data Encryption) for SINAMICS drives from firmware version V6.1.

5.23.4.20 StartDriveDownloadCheckConfiguration

StartDriveDownloadCheckConfiguration

Class `StartDriveDownloadCheckConfiguration` is derived from the class `DownloadCheckConfiguration` and has the same properties.
Class `DownloadCheckConfiguration` is described in the standard Openness help.

The class provides the configuration settings via checkboxes from the user.

The following table describes the **properties** of the class:

Name	Data type	Description
`Checked`	`bool`	Returns the current setting of the configuration or activates/ deactivates the configuration.

See also

Download (Page 1115)

5.23.4.21 SafetyTelegram

SafetyTelegram

Class `SafetyTelegram` stands for the telegram of the drive object.
Exception `EngineeringTargetInvocationException` is displayed for errors in the write attributes.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Description
`TelegramNumber`	`Int32`	Number of the main telegram. Free telegrams are designated with a value of 999.
`Type`	TelegramType (Page 1108)	Telegram type that is returned as enum "TelegramType".
`Addresses`	AddressComposition (Page 1089)	Composition of all addresses of a telegram.
`PKW`	Telegram (Page 1106)	Composition of all PKW channels of the telegram. null - if the telegram does not have a PKW part.

5.23.4.22 TechnologyExtension

TechnologyExtension

The following table describes the **properties** of the class:

Name	Data type	Access	Description
`Identifier`	`string`	ReadOnly	Returns the identifier for the Technology Extension.
`Name`	`string`	ReadOnly	Returns the name of the Technology Extension
`DisplayName`	`string`	ReadOnly	Returns the display name of the Technology Extension.
`IsActivated`	`boolean`	ReadOnly	Returns the activation status of the Technology Extension.

The following table describes the **navigators** of the class:

Name	Type	Access	Description
`Parent`	IEngineeringObject	ReadOnly	Engineering Object model-parent element of this object.

The following table describes the **methods** of the class:

Name	Return value	Parameter	Description
`Activate`	`bool`	-	Activates the Technology Extension.
`Deactivate`	`bool`	-	Deactivates the Technology Extension.

See also

Using Technology Extensions (Page 1146)

5.23.4.23 TechnologyExtensionComposition

TechnologyExtensionComposition

The `TechnologyExtensionComposition` class includes all Technology Extensions assigned to a particular DriveObject. All Technology Extensions are handled in the same way.

The following table describes the **properties** of the class:

Name	Data type	Access	Description
`Count`	`int`	ReadOnly	Returns the number of elements included in `TechnologyExtensionComposition`.

The following table describes the **navigators** of the class:

Name	Type	Access	Description
`Parent`	IEngineeringObject	ReadOnly	Engineering Object model-parent element of this object.

The following table describes the **methods** of the class:

Name	Return value	Parameter	Description
`Find`	TechnologyExtension item	string name/ identifier	Finds a Technology Extension using the name/identifier.
`Contains`	`bool`	TechnologyExtension item	Determines whether an element is included in `TechnologyExtensionComposition`.
`GetEnumerator`	IEnumerator<TechnologyExtension>	-	Returns an enumerator which runs through a list of available Technology Extensions.

5.23.4.24 TechnologyExtensionInstallationProvider

TechnologyExtensionInstallationProvider

The `TechnologyExtensionInstallationProvider` class is a function of the TIA Portal and enables a `TechnologyExtensionPackage` to be installed.

The following table describes the **navigators** of the class:

Name	Type	Access	Description
`TechnologyExtensionPackages`	TechnologyExtensionPackageComposition	ReadOnly	Compilation of `TechnologyExtensionPackages`.

The following table describes the **methods** of the class:

Name	Type of return value	Parameter	Description	Exceptions
`Install`	`bool`	ReadOnly	This method installs a `TechnologyExtensionPackage`. **Parameter description** filePath: Fully qualified path name of the .tec file to be installed.	EngineeringException
`Uninstall`	`bool`	ReadOnly	This method uninstalls a `TechnologyExtensionPackage`. Parameter description packageIdentifier: Identifier to determine the `TechnologyExtensionPackage` to be uninstalled. confirmUninstallInUse: This option is used to confirm uninstallation of a Technology Extension used in the open project.	EngineeringException

5.23.4.25 TechnologyExtensionPackage

TechnologyExtensionPackage

The `TechnologyExtensionPackage` class allows access to Technology Extensions.

The following table describes the **properties** of the class:

Name	Data type	Access	Description
`Identifier`	`string`	ReadOnly	Returns the identifier of the `TechnologyExtensionPackage`. The syntax for definition is: <name>/<version>
`Name`	`string`	ReadOnly	Returns the name of the `TechnologyExtensionPackage`.
`DisplayName`	`string`	ReadOnly	Returns the `DisplayName` of the `TechnologyExtensionPackage`.

The following table describes the **navigators** of the class:

Name	Type	Access	Description
`Parent`	IEngineeringObject	ReadOnly	Engineering Object model-parent element of this object.

5.23.4.26 Telegram

Telegram

Class `Telegram` allows access to the structure of a telegram from a drive object.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public sealed class Telegram
```

The following table describes the **properties** of the class:

Name	Data type	Description
`TelegramNumber`	`Int32`	Returns the number of the main telegram or specifies the number. A freely configurable telegram has the number 999.
`Type`	TelegramType (Page 1108)	Returns the type of telegram as `Enum TelegramType`.
`Addresses`	AddressComposition (Page 1089)	Returns an `AdressComposition` with information on the address.
`PKW`	`Telegram`	Returns channel `PKW` as `Telegram`. `null` if the property is not available A telegram with `PKW` is telegram 353, for example.
`HardwareIdentifier`	`IEnumerable<IBrowsable>`	Returns the list of hardware identifiers.

The following table describes the **methods** of the class:

Name	Description
`CanChangeTelegram(Int32)`	Returns `true` if the telegram can be changed to the parameterized standard type.
`GetSize(`AddressIoType (Page 1090)`)`	Returns the size of the inputs or outputs of the telegram.

Name	Description
`CanChangeSize(`AddressIoType (Page 1090)`, Int32, bool)`	Returns `true` if the size of the telegram can be changed as parameterized. Standard telegrams can only be enlarged. The retention of the previous telegram address is taken into account when the option is parameterized with `true`.
`ChangeSize(`AddressIoType (Page 1090)`, Int32, bool)`	Returns `true` if the size of the telegram could be changed as parameterized. The retention of the previous telegram address is taken into account when the option is parameterized with `true`.

5.23.4.27 TelegramComposition

TelegramComposition

The `TelegramComposition` class allows access to the telegrams of a drive object. The structure of a telegram can be read out via class Telegram (Page 1106).

Note that referenced objects may become invalid through class `TelegramComposition`. For example, object Telegram (Page 1106) becomes invalid after the telegram size is changed.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public sealed class TelegramComposition
```

The following table describes the **methods** of the class:

Name	Description
`CanInsertAdditionalTelegram(Int32, Int32)`	Returns `true` if an extension can be created in accordance with the parameterized sizes (input and output sizes).
`CanInsertTorqueTelegram (Int32, telegramNumber)`	When adding a new torque telegram with an already existing telegram number, checks whether this is possible.
`CanInsertSupplementaryTelegram(Int32)`	Returns `true` if a supplementary telegram can be created in accordance with the parameterized telegram number.
`EraseTelegram(TelegramType)`	Deletes a telegram with a known telegram type from the drive object. Returns `true` if the parameterized telegram could be deleted. If a torque telegram is used as type, then object "torqueTelegram" is deleted. If a Safety Integrated telegram is used as type, then object "safetyTelegram" is deleted. Standard telegrams cannot be deleted. In the event of an error, an `EngineeringTargetInvocationException` is initiated.

Name	Description
`Find(TelegramType)`	Returns the Telegram (Page 1106) object if it could be found via the parameterized telegram type or Safety Integrated telegram type. `null` if the telegram is not found. If a torque telegram is used as type, then object `torqueTelegram` is returned, assuming that it exists. If a Safety Integrated telegram is used as type, then object `safetyTelegram` is returned, assuming that it exists. **Example** `Telegram telegram = telegrams.Find(TelegramType.MainTelegram);`
`InsertAdditionalTelegram(Int32, Int32)`	Creates an extension for the drive object in accordance with the parameterized sizes and returns `true` if the extension could be inserted. In the event of an error, an `EngineeringTargetInvocationException` is initiated.
`InsertTorqueTelegram (Int32, telegramNumber)`	Adds a new torque telegram with an already existing telegram number to a drive object.
`InsertSafetyTelegram (Int32, telegramNumber)`	Adds a Safety Integrated telegram with its specified telegram number to a drive object.
`InsertSupplementaryTelegram(Int32)`	Creates the supplementary telegram with the parameterized telegram number and returns `true` if the telegram could be inserted. In the event of an error, an `EngineeringTargetInvocationException` is initiated.

5.23.4.28 TelegramType

TelegramType

The Enum TelegramType contains predefined telegram types.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **syntax** of the class:

```
public enum TelegramType
```

The following table describes the **enum entries**:

Name	Description
`MainTelegram`	ID of the main telegram
`SupplementaryTelegram`	ID of the supplementary telegram
`AdditionalTelegram`	ID of an extension

5.23.4.29 TorqueTelegram

TorqueTelegram

Class `TorqueTelegram` stands for the telegram of the drive object.
Exception `EngineeringTargetInvocationException` is displayed for errors in the write attributes.

Namespace: `Siemens.Engineering.MC.Drives`
Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Description
`TelegramNumber`	`Int32`	Telegram number.
`Type`	TelegramType (Page 1108)	Telegram type that is returned as enum "TelegramType".
`Addresses`	AddressComposition (Page 1089)	Composition of all addresses of a telegram.
`PKW`	Telegram (Page 1106)	Composition of all PKW channels of the telegram. null - if the telegram does not have a PKW part.

5.23.4.30 AxisEncoderHardwareConnectionInterface

AxisEncoderHardwareConnectionInterface

The class facilitates access to the properties of the technology object.

The following table describes the **methods** of the class:

Name	Parameter	Return value	Description
`Connect`	Telegram	bool	Defines the non-DRIVE-CLiQ encoder type to be either rotary or linear.
`Connect`	Telegram, ConnectOption	bool	Defines the DRIVE-CLiQ encoder type to be either rotary or linear as well as absolute or incremental. Not intended for use with non-DRIVE-CLiQ encoders. G220 drives with firmware version V6.2 only support absolute or incremental encoder types. This API can be used for setting.

5.23.4.31 UmacConfiguration (from SINAMICS FW V6.1)

For SINAMICS drives from firmware version V6.1, class UmacConfiguration allows function User Management and Access Control (UMAC) to be activated and deactivated with Openness.

The following table describes the **methods** of class `UmacConfiguration`:

Name	Parameter	Return value	Description	Exceptions
`Activate`	-	bool	Activates `UmacConfiguration` for the drive.	-
`Deactivate`	-	bool	Deactivates `UmacConfiguration` for the drive.	-

Note

`Security` class

The `Security` class is a higher-level class. The class is used in the background to access all APIs that are involved in security.

5.23.4.32 UpdateCheckSums (from SINAMICS FW V6.1)

For SINAMICS drives from firmware version V6.1, class `UpdateCheckSums` allows Safety Integrated Functions to be configured with Openness.

The following table describes the **methods** of class `UpdateCheckSums`:

Name	Parameter	Return value	Description	Exceptions
`UpdateCheckSums`	-	bool	Recalculates and overwrites the checksum value of Safety Integrated reference parameters.	-

5.23.5 Code examples

The following code examples describe the basic procedure for various applications. The code is not necessarily complete or compilable.

5.23.5.1 Determining the activation status

The following examples show how you can determine the activation status for S120 drives, either offline or online:

Determining the activation status of an S120 drive offline

```
using Siemens.Engineering.MC.Drives;
```

Determining the activation status of an S120 drive offline

```
DriveFunctionInterface dfi = ...
DriveObjectActivation driveObjectActivation =
dfi.DriveObjectFunctions.DriveObjectActivation;
//driveObjectActivation can be null in case of the actual driveobject does not support
activation.

//change activation state
driveObjectActivation.ChangeActivationState(DriveObjectActivationState.Deactivate);

//get the activation state
DriveObjectActivationState activationState = driveObjectActivation.ActivationState;

//get the Is Active property
bool isActive = driveObjectActivation.IsActive;
```

Determining the activation status of an S120 drive online

```
using Siemens.Engineering.MC.Drives;

OnlineDriveFunctionInterface onlinedfi = ...
DriveObjectActivation driveObjectActivation = onlinedfi.DriveObjectActivation;
//driveObjectActivation can be null in case of the actual driveobject does not support
activation in online.

//change activation state
driveObjectActivation.ChangeActivationState(DriveObjectActivationState.Deactivate);

//get the activation state
DriveObjectActivationState activationState = driveObjectActivation.ActivationState;

//get the IsActive property
bool isActive = driveObjectActivation.IsActive;
```

5.23.5.2 Executing drive functions

The following examples show how you can simply access drive functions, either offline or online:

Executing drive functions offline

```
using Siemens.Engineering.MC.Drives;

DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();

// dfi can be null in case of the actual driveobject does not support it.
```

Executing drive functions online

```
using Siemens.Engineering.MC.Drives;

OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
```

5.23.5.3 Creating a drive

You can create a drive with method `CreateWithItem()` of collection `Devices`.

The drives are specified using parameters of the method. The format of the parameters is described in the following.

Creating a G120 drive

Format of the parameters for a G120:

```
CreateWithItem(@"OrderNumber: mlfb / FirmwareVersion /",
"NameOfTheDevice", positionNumber)
```

The following example shows how you can create a G120 drive.

Creating a G120 drive

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open("..."); //The path of the project

Device s120Device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3246-0BA22-1FA0/4.7.6/"
, "Device_0", null);
```

Creating S120, S150, S200, S210, MV, G130, G150 and G220 drives

Parameter format for S120, S150, S200, S210, MV, G130, G150 and G220 drives:

```
CreateWithItem(@"OrderNumber: mlfb / FirmwareVersion /
AdditionalTypeIdentifier", "NameOfTheDevice", positionNumber)
```

Possible values for `AdditionalTypeIdentifier`:

- Empty string (e.g. for G120)
- S120
- S150
- S200
- S210
- MV
- G130
- G150
- G220

The following example shows how you can create an S120 drive.

Creating an S120 drive

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open("..."); //The path of the project

Device s120Device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3040-1MA01-0Axx/V4.8/
S120", "Device_0", null);
```

5.23.5.4 Creating a drive component

You can create a drive component for a drive unit with the `PlugNew()` method of the `Device` object.

The following example shows how to create a drive component.

Creating a Motor Module

```
DeviceItem subModul = sdrDevice.PlugNew(@"OrderNumber:6SL3xxx-xxxxx-xxxx",
"MotorModul", 65535);
```

5.23.5.5 Determining a drive object

The following examples show how to determine drive objects offline and online.

Determining an offline drive object

```
using Siemens.Engineering.MC.Drives;
//G devices
Project project =  portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0].Items[0];
DriveObject driveObject =
item.GetService<DriveObjectContainer>().DriveObjects[0];

//S devices
Project project =  portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
DriveObject driveObject =
item.GetService<DriveObjectContainer>().DriveObjects[0];
```

Determining an online drive object

```
using Siemens.Engineering.MC.Drives;
```

Determining an online drive object

```
//G devices
Project project =  portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0].Items[0];
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

//S devices
Project project =  portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
```

5.23.5.6 Determining the drive object type

The following examples shows how you can determine the actual drive object type – and the types that can be alternatively set.

Determining drive object types

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;

DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();
DriveObjectTypeHandler driveObjectTypeHandler =
dfi.DriveObjectFunctions.DriveObjectTypeHandler;

// dfi can be null in case of the actual driveobject does not support it.

// driveObjectTypeHandler can be null, if the actual driveObject does not support it.

// Get the possible drive object types on the drive object.
DriveObjectTypeComposition possibleDriveObjectTypes =
driveObjectTypeHandler.PossibleDriveObjectTypes;

// Get the current drive object type on the drive object.
DriveObjectType currentDriveObjectType = driveObjectTypeHandler.CurrentDriveObjectType;

//Call the ChangeDriveObjectType method with the current drive object type.
//The method parameter should be the target drive object type.
driveObjectTypeHandler.ChangeDriveObjectType(possibleDriveObjectTypes[0]);
```

5.23.5.7 Reading and writing BICO parameters

The following example shows how to read and write values of BICO parameters. You require a drive object for access.

Reading BICO parameters

```
using Siemens.Engineering.MC.Drives;
```

Reading BICO parameters

```
DriveParameter bicoSink= driveObject.Parameters.Find("p681");
if(bicoSink!=null)
{
  if(bicoSink.Value is DriveParameter)
  {
    DriveParamter bicoSourceValue = bicoSink.Value as DriveParameter;
    Console.WriteLine("The value of parameter " + bicoSink.Name + ": " +
bicoSource.Name + " " + bicoSource.ParameterText);
  }
  else if (bicoSink.Value == null)
  {
    Console.WriteLine("Value contains an invalid connection or the source
parameter is not accessible via Openness");
  }
  else
  {
    Console.WriteLine("The value of parameter " + bicoSink.Name + ": " +
bicoSink.Value.ToString());
  }
}
```

Writing BICO parameters

```
using Siemens.Engineering.MC.Drives;
DriveParameter bicoSource= driveObject.Parameters.Find("r19");
DriveParameter bicoSink = driveObject.Parameters.Find("p738");
if(bicoSource != null)
{
  try
  {
    bicoSink.Value = bicoSource;
  }
  catch(UserException ex)
  {
    Console.WriteLine("Write failure :" + ex.Message);
  }
}
```

5.23.5.8 Download

After starting the download, you must adjust and confirm the configuration settings. The configuration settings are provided as child objects of object `DownloadConfiguration` – and there are three different types:

- `StartDriveDownloadCheckConfiguration`
- `DownloadSelectionConfiguration`
- `DownloadPasswordConfiguration`

The following examples show the evaluation of the different types of configuration settings in the PreDownload Delegate.

Evaluation of the configuration settings after starting the download

```
using Siemens.Engineering.Download;
using Siemens.Engineering.Online;
static void PreDownload(DownloadConfiguration configuration)
{
  Console.WriteLine(configuration.Message);
  StartDriveDownloadCheckConfiguration sdcc = configuration as
StartDriveDownloadCheckConfiguration;
  if (sdcc != null)
  {
    sdcc.Checked = true;
    return;
  }

  DownloadPasswordConfiguration downloadPasswordConfiguration =
configuration as DownloadPasswordConfiguration;

  if (downloadPasswordConfiguration != null)
  {
    SecureString s = new SecureString();
    string passwordText = "password";
    foreach (var str in passwordText)
    {
      s.AppendChar(str);
    }

    downloadPasswordConfiguration.SetPassword(s);
    return;
  }

  DownloadSelectionConfiguration downloadSelectionConfiguration =
configuration as DownloadSelectionConfiguration;

  if (downloadSelectionConfiguration != null)
  {
    downloadSelectionConfiguration.SelectedIndex = 0;
    return;
  }
}
```

The following examples show how to download a project to the device.

Download to the S120 device

```
using Siemens.Engineering.Download;
using Siemens.Engineering.Online;
```

Download to the S120 device

```
try
{
  DeviceItem item = ... //device item of the CU (e.g. :
project.Devices[0].Items[0].Items[0])
  DownloadProvider downloadProvider = item.GetService<DownloadProvider>();

  DownloadConfigurationDelegate pre = PreDownload;
  DownloadConfigurationDelegate post = PostDownload;

  ConnectionConfiguration connConfiguration =
downloadProvider.Configuration;
  ConfigurationMode configurationMode = connConfiguration.Modes.Find("PN/
IE");
  ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces[0];
  ConfigurationSubnet subnet = pcInterface.Subnets.Find(/*subnet name*/);
  IConfiguration configuration = subnet.Addresses.Find(/*IP address of the
device*/);
  downloadProvider.Download(configuration, pre, post,
DownloadOptions.Software);
}
catch (Exception e)
{
  Console.WriteLine(e.Message);
}

//configuration handling before download
static void PreDownload(DownloadConfiguration configuration)
{
  Console.WriteLine(configuration.Message);
  StartDriveDownloadCheckConfiguration dcc = configuration as
StartDriveDownloadCheckConfiguration ;
  if (dcc != null)
  {
    dcc.Checked = true;
  }
}

//configuration handling after download
static void PostDownload(DownloadConfiguration configuration)
{
  Console.WriteLine(configuration.Message);
}
```

Additional settings for downloads for drives as of SINAMICS FW V6.1

If you add a drive with SINAMICS FW V6.1 or higher to a project, then the security settings for this drive are in the factory setting.

It is only possible to download from a drive such as this if UMAC is activated for the project.

Configuring UMAC

```
UmacConfigurator UmacConfiguratorService =
project.GetService<UmacConfigurator>();var projectUsers =
UmacConfiguratorService.ProjectUsers;
```

Creating a user for the project

```
ProjectUser projectUser = projectUsers.Create("AdminUser",
"AdminUser123#".ToSecureString());
```

Adding a system role

```
projectUser.Roles.Add(systemRole);
```

Also requires that a valid TLS certificate is downloaded as described in the following example:

Setting a TLS certificate

```
private static OnlineConfigurationDelegate m_OnlineConfigurationDelegate =
null;
private bool TryDownloadToDeviceTest()
 {
 String newAddress = ...// network ip;
 try
 {
 Device device =...//device
 DeviceItem item = device.Items[0].Items[0];
 DownloadProvider downloadProvider = item.GetService<DownloadProvider>();
 ConnectionConfiguration conf = downloadProvider.Configuration;
}
}
```

Activating notification for TLS events

```
 if (m_OnlineConfigurationDelegate == null)
 {
 m_OnlineConfigurationDelegate = OnlineCallBackMethod;
 conf.OnlineLegitimation += m_OnlineConfigurationDelegate;
 }
```

Setting download configuration settings

```
//perform download
{
catch (Exception e)
        {
            Console.WriteLine(e.ToString());
            return false;
        }
        return true;
        }
    private static void OnlineCallBackMethod(OnlineConfiguration
onlineConfiguration)
        {
            TlsVerificationConfiguration verificationConfiguration =
onlineConfiguration as TlsVerificationConfiguration;
            if (verificationConfiguration != null)
            {
                verificationConfiguration.CurrentSelection =
TlsVerificationConfigurationSelection.Trusted;
            }
}
```

See also

Encrypted communication with SecureString passwords (Page 1087)

5.23.5.9 Editing DRIVE-CLiQ connections

The following example shows how to edit DRIVE-CLiQ connections with Openness.

Editing DRIVE-CLiQ connections

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
```

Editing DRIVE-CLiQ connections

```
Project tiaproject = portal.Projects.Open(new
FileInfo(@"C:\Users\testUser\Documents\Automation\Project109\Project109.ap
15")); //The path of the project

Device device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3040-1MA01-0Axx/V4.8/
S120", "Device_0", null); //Create the device

DeviceItem subModul = device.PlugNew(@"OrderNumber:6SL3x2x-1xxxx-xxxx",
"", 65535); //Plug a submodul (in this case : Motor modul)

DeviceItem cu = device.DeviceItems[1];          // The CU is always the 1
indexed in the DeviceItems of the Device

DeviceItem subModulDQInterface = subModul.DeviceItems[0];      //We need
the DQ interface from the submodul
DeviceItem cuDQInterface = cu.DeviceItems[3];                //This is the
DQ interface of the CU

NetworkPort subModulDQ =
((IEngineeringServiceProvider)subModulDQInterface.DeviceItems[0]).GetServi
ce<NetworkPort>(); //We need the DriveCliq port of the DQ interface from
the submodul
NetworkPort cuDQ =
((IEngineeringServiceProvider)cuDQInterface.DeviceItems[0]).GetService<Net
workPort>();               //We need the DriveCliq port of the DQ interface
from the CU

cuDQ.DisconnectFromPort(subModulDQ); //Delete the connection between the
two ports (automatically created when plugging a modul)
cuDQ.ConnectToPort(subModulDQ); //Create a new connection
```

5.23.5.10 Carrying out the first steps in Startdrive

The following example shows how you can localize or generate an active TIA Portal process.

Finding or generating a TIA Portal process

```
using Siemens.Engineering;

// Get the list of the running TIA Portal processes.
IList<TiaPortalProcess> procs = TiaPortal.GetProcesses();
TiaPortal portal;
// When there is at least one running TIA Portal, we will attach to the first from the list.
if (procs.Count != 0)
{
  portal = procs[0].Attach();
}
// When there is no running TIA Portal, we create one.
else
{
  portal = new TiaPortal(TiaPortalMode.WithUserInterface);
}
```

The following example shows how you can localize or generate a Startdrive project.

Finding or generating a Startdrive project

```
using Siemens.Engineering;

Project project;
// When the portal has one project, we save it in a variable.
if (portal.Projects.Count == 1)
{
  project = portal.Projects[0];
}
// When there is no existing project, we create one with a specific path, and the actual time
else
{
  project = portal.Projects.Create(
  new DirectoryInfo(@"C:\Projects\Project_" + DateTime.Now.Ticks),
  DateTime.Now.Ticks.ToString());
}
```

The following example shows how you can determine as to whether a specific Startdrive variant (package and version) is installed.

Determining whether the required Startdrive version is installed

```
using Siemens.Engineering;

if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINAMICS Startdrive Advanced")
&& sw.Version.Equals("V15"))) { Console.WriteLine("Startdrive is available");}
// "V15" is the current startdrive version started at December 2017.
// "V15.1" will be the current startdrive version beginning with December 2018.
// "SINAMICS Startdrive Basic" and "SINAMICS Startdrive Advanced" are the 2 possible
startdrive function packages.
```

5.23.5.11 Defining the encoder type

The following example shows how you can set the encoder type or the encoder data set number offline.

Setting the encoder type and/or the encoder data set number offline via the hardware configuration

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;
```

Setting the encoder type and/or the encoder data set number offline via the hardware configuration

```
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];

DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// To enable the setting of an EncoderType, the value of p96 should
// be set to 0 (Application class == [0] Expert) if it is present on
// the current G drive.
DriveParameter p96 = cuDriveObject.Parameters.Find("p96");
if (p96 != null)
{
  p96.Value = 0;
}

// Setting Encoder 1 on the drive.
// In case of encoders, we have to set several enums to define an
// encoder type. These enums are: EncoderInterface, EncoderType,
// AbsoluteIncrementalFlag, and RotaryLinearFlag.

// There can be a problem, if the given enum combination is not valid.
// In that case, it has to give back a feedback.

hardwareProjection.SetEncoder(
  EncoderInterface.Terminal,
  EncoderType.HTLTTL,
  AbsoluteIncrementalFlag.Incremental,
  RotaryLinearFlag.Rotary,
  1);

// It is possible to set 2 encoders to a motor, one with encoderNumber == 1
// and the other with encoderNumber == 2
```

The following example shows how you can set the encoder type or the encoder data set number using an online connection.

Setting the encoder type and/or the encoder data set number online via the hardware configuration

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;
```

Setting the encoder type and/or the encoder data set number online via the hardware configuration

```
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
OnlineDriveObject cuOnlineDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface cuDriveFunctionInterface =
cuOnlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// To enable the setting of an Encoder, the value of p96 should
// be set to 0 (Application class == [0] Expert) if it is present
// the current G drive.
DriveParameter p96 = cuOnlineDriveObject.Parameters.Find("p96");
if (p96 != null)
{
  p96.Value = 0;
}

// Setting Encoder 1 on the drive.
// In case of encoders we have to set several enums to define an
// encoder type. These enums are: EncoderInterface, EncoderType,
// AbsoluteIncrementalFlag, and RotaryLinearFlag.

// There can be a problem, if the given enum combination is not valid.
// In that case, it has to give back a feedback.

hardwareProjection.SetEncoder(
  EncoderInterface.Terminal,
  EncoderType.HTLTTL,
  AbsoluteIncrementalFlag.Incremental,
  RotaryLinearFlag.Rotary,
  1);

// It is possible to set 2 encoders to a motor, one with encoderNumber == 1
// and the other with encoderNumber == 2
```

The following example shows how you can read out the actual encoder configuration from the drive.

Reading out the encoder configuration

```
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();

HardwareProjection encoderProjection = cuDriveFunctionInterface.HardwareProjection;

// Before setting an encoder, p96 should be 0
// (See section "Projecting EncoderConfiguration")

encoderProjection.SetEncoder(
  EncoderInterface.Terminal,
  EncoderType.HTLTTL,
  AbsoluteIncrementalFlag.Incremental,
  RotaryLinearFlag.Rotary,
  1);

EncoderConfiguration encoderConfiguration =
encoderProjection.GetCurrentEncoderConfiguration(1);
```

The following example shows how you can configure the actual encoder configuration offline.

Configuring an encoder offline

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Configuring an encoder offline

```
// Project encoder configuration in Offline state
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];

DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// Before setting an encoder type, p96 should be 0
// (See section "Projecting EncoderConfiguration")

// To Project an EncoderConfiguration, first set e.g. Encoder 1 with .SetEncoder(...).

hardwareProjection.SetEncoder(
  EncoderInterface.Terminal,
  EncoderType.HTLTTL,
  AbsoluteIncrementalFlag.Incremental,
  RotaryLinearFlag.Rotary,
  1);

// Get the current configuration of Encoder 1.
EncoderConfiguration config = hardwareProjection.GetCurrentEncoderConfiguration(1);

// Fill out Encoder 1's configuration values.
config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
  switch (ce.Name)
  {
    case "p405.0":
      ce.Value = 0;
      break;
    case "p405.1":
      ce.Value = 1;
      break;
    case "p408":
      ce.Value = 1024;
      break;
    case "p425":
      ce.Value = 2048;
      break;
    default:
      break;
  }
});

// Project the Encoder 1 configuration to the device.
bool result =
cuDriveFunctionInterface.HardwareProjection.ProjectEncoderConfiguration(config, 1);
```

The following example shows how you can configure the actual encoder configuration online.

Configuring an encoder online

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Configuring an encoder online

```
// Project encoder configuration in Online state
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DeviceItem cuDeviceItemForOnline = m_Device.DeviceItems[0];
//You need the rack for going online on G120 Device, which is .DeviceItems[0]

// GoOnline...
// To use these function in Online, you have to use the OnlineDriveFunctionInterface
OnlineDriveObject onlineDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// Before setting an encoder, p96 should be 0
// (See section "Projecting EncoderConfiguration")

// To Project an EncoderConfiguration, first set e.g. Encoder 1 with .SetEncoder(...).

hardwareProjection.SetEncoder(
  EncoderInterface.Terminal,
  EncoderType.HTLTTL,
  AbsoluteIncrementalFlag.Incremental,
  RotaryLinearFlag.Rotary,
  1);

// Get the current configuration of Encoder 1.
EncoderConfiguration config = hardwareProjection.GetCurrentEncoderConfiguration(1);

// Fill out Encoder 1's configuration values.
config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
  switch (ce.Name)
  {
    case "p405.0":
      ce.Value = 0;
      break;
    case "p405.1":
      ce.Value = 1;
      break;
    case "p408":
      ce.Value = 1024;
      break;
    case "p425":
      ce.Value = 2048;
      break;
    default:
      break;
  }
});

// Project the Encoder 1 configuration to the device.
bool result = onlineDfi.HardwareProjection.ProjectEncoderConfiguration(config, 1);
```

The following example shows how you can write the configuration entry to the console.

Write out configuration entry

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
MotorConfiguration motorConfig = hardwareProjection.GetCurrentMotorConfiguration(0);

foreach (var configurationEntry in motorConfig.RequiredConfigurationEntries)
{
  Console.WriteLine(configurationEntry.Name);
}
EncoderConfiguration encConfig = hardwareProjection.GetCurrentEncoderConfiguration(1);

foreach (var configurationEntry in encConfig.RequiredConfigurationEntries)
{
  Console.WriteLine(configurationEntry.Name);
}
```

The following example shows how you can configure DRIVE-CLiQ encoders.

Configuring DRIVE-CLiQ encoders

```
using Siemens.Engineering.MC.Drives.Enums;
```

Configuring DRIVE-CLiQ encoders

```
private bool TestEncoderProjectionForAcxDrives(){
    bool result = false;
    //Revelant drive already added to project m_Project.Devices[0]
   DeviceItem motorModule = m_Project.Devices[0].DeviceItems[2]; //get the motor module or
power module
    DriveObject mmDriveObject =
motorModule.GetService<DriveObjectContainer>().DriveObjects[0];
    DriveFunctionInterface mmDriveFunctionInterface =
mmDriveObject.GetService<DriveFunctionInterface>();
    HardwareProjection hwProjection = mmDriveFunctionInterface.HardwareProjection;

    try
    {
        EncoderConfiguration encoderConfiguration =
hwProjection.GetCurrentEncoderConfiguration(0);
        //get EncoderConfiguration for encoder at index 0

        IList<ConfigurationEntry> encoderTypes =
encoderConfiguration.EncoderTypes.ToList();
        //get the type of encoder, this would return only the applicable parameters p404.0
& p404.1

        bool setEncoderTypeResult =
encoderConfiguration.SetEncoderType(RotaryLinearFlag.Linear,
AbsoluteIncrementalFlag.Incremental);
        //SetEncoderType is optional and can be used to change Encoder Type.
        //SetEncoderType can be used to set Rotary or Linear as Encoder Type.
       //For DriveCliq encoder, along with Rotary/Linear, you can also set the Encoder type
as Incremental or Absolute.

        if (setEncoderTypeResult)
        {
            encoderConfiguration.RequiredConfigurationEntries.ToList().ForEach(e =>
            {
                if (e.Name.Equals("p404.0"))
                {
                    e.Value = true;
                }
            }); //set parameter p404.0 (Linear encoder) to true
            result = hwProjection.ProjectEncoderConfiguration(encoderConfiguration, 0);
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message+ "\n" + e.StackTrace);
    }
    return result;
}
```

The following example shows how you can configure non-Siemens encoders without DRIVE-CLiQ.

Configuring non-Siemens encoders without DRIVE-CLiQ

```
using Siemens.Engineering.MC.Drives.Enums;
```

Configuring non-Siemens encoders without DRIVE-CLiQ

```
private bool TestEncoderProjectionForAcxDrives()
{
    bool result = false;
    //Revelant drive already added to project m_Project.Devices[0]
    DeviceItem motorModule = m_Project.Devices[0].DeviceItems[2]; //get the motor module or
power module
    DriveObject mmDriveObject =
motorModule.GetService<DriveObjectContainer>().DriveObjects[0];
    DriveFunctionInterface mmDriveFunctionInterface =
mmDriveObject.GetService<DriveFunctionInterface>();
    HardwareProjection hwProjection = mmDriveFunctionInterface.HardwareProjection;

    try
    {
        EncoderConfiguration encoderConfiguration =
hwProjection.GetCurrentEncoderConfiguration(0);
        //get EncoderConfiguration for encoder at index 0

        IList<ConfigurationEntry> encoderTypes =
encoderConfiguration.EncoderTypes.ToList();
        //get the type of encoder, this would return only the applicable parameter p404.0

        bool setEncoderTypeResult =
encoderConfiguration.SetEncoderType(RotaryLinearFlag.Linear);
        //SetEncoderType is optional and can be used to change Encoder Type.
        //SetEncoderType can be used to set Rotary or Linear as Encoder Type.

        if (setEncoderTypeResult)
        {
            encoderConfiguration.RequiredConfigurationEntries.ToList().ForEach(e =>
            {
                if (e.Name.Equals("p404.0"))
                {
                    e.Value = true;
                }
            }); //set parameter p404.0 (Linear encoder) to true
            result = hwProjection.ProjectEncoderConfiguration(encoderConfiguration, 0);
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message+ "\n" + e.StackTrace);
    }
    return result;
}
```

5.23.5.12 Configuring devices

The following example shows how you can configure S120 and G120 drives offline.

Configuring S120 and G120 drives offline

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;
```

Configuring S120 and G120 drives offline

```
DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = dfi.HardwareProjection;

// dfi can be null in case of the actual driveobject does not support it.
// hardwareProjection can be null, if the actual driveObject does not support it.
// For example: On G120 drives, you have to use the CU as driveobject. On S120
// drives, you have to use the MotorModul as driveObject.
```

The following example shows how you can configure S120 and G120 drives online.

Configuring S120 and G120 drives online

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;

OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
// hardwareProjection can be null, if the actual onlineDriveObject
// does not support it.
// For example: On G120 drives, you have to use the CU as onlineDriveObject. On
// S120 drives, you have to use the MotorModul as onlineDriveObject.
```

5.23.5.13 Determining hardware identifiers

The following examples show how you can determine the hardware identifier (Page 1106) of various objects.

Determining the hardware identifiers of telegrams

```
using Siemens.Engineering.MC.Drives;

Device s120Drive = m_Project.Devices[1];
DeviceItem driveUnit = s120Drive.DeviceItems[0];
DriveObjectContainer driveObjectContainer =
driveUnit.GetService<DriveObjectContainer>();
DriveObject driveObject = driveObjectContainer.DriveObjects[0];
Telegram mainTelegram = driveObject.Telegrams[0];

IList<HwIdentifier> hwIdentifiers =
mainTelegram.HwIdentifiers?.ToList(); //get the Hardware Identifier
long identifier = hwIdentifiers[0].Identifier;
```

Determining the hardware identifier of a Module Access Point

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.HW.HwIdentifier;
```

Determining the hardware identifier of a Module Access Point

```
Device s120Drive = m_Project.Devices[1];
DeviceItem driveUnit = s120Drive.DeviceItems[0];
DriveObjectContainer driveObjectContainer =
driveUnit.GetService<DriveObjectContainer>();
DriveObject driveObject = driveObjectContainer.DriveObjects[0];
Telegram mainTelegram = driveObject.Telegrams[0];

IList<HwIdentifier> hwIdentifiers =
mainTelegram.HwIdentifiers?.ToList(); //get the Hardware Identifier
long identifier = hwIdentifiers[0].Identifier;

Device deviceS120 = m_Project.Devices[1]; //S120 with Double Motor Module
Device deviceG120 = m_Project.Devices[2]; //G120
Device deviceG220 = m_Project.Devices[3]; //G220

ModuleAccessPoint moduleAccessPointS120 =
deviceS120.DeviceItems[1].GetService<ModuleAccessPoint>();
IList<HwIdentifier> hwIdentifiersS120 =
moduleAccessPointS120.HwIdentifiers.ToList();
long identifierS120 = hwIdentifiersS120[0].Identifier;

ModuleAccessPoint moduleAccessPointDriveAxis =
deviceS120.DeviceItems[2].GetService<ModuleAccessPoint>();
IList<HwIdentifier> hwIdentifiersDriveAxis =
moduleAccessPointDriveAxis.HwIdentifiers.ToList();
long identifierDriveAxis = hwIdentifiersDriveAxis[0].Identifier;

ModuleAccessPoint moduleAccessPointDriveAxisTwo =
deviceS120.DeviceItems[3].GetService<ModuleAccessPoint>();
IList<HwIdentifier> hwIdentifiersDriveAxisTwo =
moduleAccessPointDriveAxisTwo.HwIdentifiers.ToList();
long identifierDriveAxisTwo = hwIdentifiersDriveAxisTwo[0].Identifier;

ModuleAccessPoint moduleAccessPointG120 =
deviceG120.DeviceItems[1].GetService<ModuleAccessPoint>();
IList<HwIdentifier> hwIdentifiersG120 =
moduleAccessPointG120.HwIdentifiers.ToList();
long identifierG120 = hwIdentifiersG120[0].Identifier;

ModuleAccessPoint moduleAccessPointG220 =
deviceG220.DeviceItems[1].GetService<ModuleAccessPoint>();
IList<HwIdentifier> hwIdentifiersG220 =
moduleAccessPointG220.HwIdentifiers.ToList();
long identifierG220 = hwIdentifiersG220[0].Identifier;
```

5.23.5.14 Creating a component for a drive component (S120 only)

You have the option of creating a component below a drive component for the S120.

Also enter a type designation to distinguish between the various encoders. The possible type designations and restrictions for encoders are described in the table below.

The following example shows how to create a component below a drive component.

Creating a motor and an encoder below a Motor Module

```
DeviceItem subModul = sdrDevice.PlugNew(@"OrderNumber:6SL3xxx-xxxxx-xxxx",
"MotorModul", 65535);

//Plug a motor to the motor modul
subModul.Container.PlugNew(@"OrderNumber:1PH2092-4WG4x-xxxx",
"Motor_1",65535);

//Plug an encoder to the motor modul
subModul.Container.PlugNew(@"OrderNumber:XExxxxx-xxxxx-xxxx//DRIVE-
CLIQ.202", "Encoder_1",65535);
```

Type designations for encoders and restrictions

The following restrictions apply when inserting encoders via Openness:

- Only an unspecific Sensor Module can be created for some encoders when inserting via Openness. In this case, you must configure the specific type of the Sensor Module in the TIA Portal.
- Maximum two encoders can be inserted for one Motor Module.

The following table lists the available type designations for encoders.

DRIVE-CLiQ	Resolver	sin/cos	SSI	sin/cos+SSI	HTL/TTL	HTL/TTL+SSI	EnDat 2.1
DRIVE-CLIQ.202	Resolver.0	SIN_COS.0	SSI.0	SIN_COS_+_SSI.0	HTL_TTL.0	HTL_TTL+SSI.0	EnDat_2.1.2051
DRIVE-CLIQ.204	Resolver.1001	SIN_COS.2001	SSI.3081	SIN_COS_+_SSI.2081	HTL_TTL.3001	HTL_TTL+SSI.3088	EnDat_2.1.2052
DRIVE-CLIQ.212	Resolver.1002	SIN_COS.2002	SSI.3082	SIN_COS_+_SSI.2082	HTL_TTL.3002	HTL_TTL+SSI.3090	EnDat_2.1.2053
DRIVE-CLIQ.214	Resolver.1003	SIN_COS.2003	SSI.9999	SIN_COS_+_SSI.2083	HTL_TTL.3003	HTL_TTL+SSI.9999	EnDat_2.1.2054
DRIVE-CLIQ.242	Resolver.1004	SIN_COS.2004	-	SIN_COS_+_SSI.2084	HTL_TTL.3005	-	EnDat_2.1.2055
DRIVE-CLIQ.244	Resolver.9999	SIN_COS.2005	-	SIN_COS_+_SSI.9999	HTL_TTL.3006	-	EnDat_2.1.2151
DRIVE-CLIQ.9999	-	SIN_COS.2006	-	-	HTL_TTL.3007	-	EnDat_2.1.9999
DRIVE-CLIQ.10100	-	SIN_COS.2007	-	-	HTL_TTL.3008	-	EnDat_2.1.10100
-	-	SIN_COS.2008	-	-	HTL_TTL.3009	-	-
-	-	SIN_COS.2010	-	-	HTL_TTL.3011	-	-
-	-	SIN_COS.2012	-	-	HTL_TTL.3020	-	-
-	-	SIN_COS.2013	-	-	HTL_TTL.3109	-	-

DRIVE-CLiQ	Resolver	sin/cos	SSI	sin/cos+SSI	HTL/TTL	HTL/TTL+SSI	EnDat 2.1
-	-	SIN_COS.2110	-	-	HTL_TTL.9999	-	-
-	-	SIN_COS.2111	-	-	-	-	-
-	-	SIN_COS.2112	-	-	-	-	-
-	-	SIN_COS.9999	-	-	-	-	-

5.23.5.15 Defining the motor type and motor configuration

The following examples show how you can set a motor type for G120 drives via the device configuration:

Setting the motor type offline via the hardware configuration

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

//Offline (Only on G120 drives)
DriveFunctionInterface dfi = ...
HardwareProjection hardwareProjection = dfi.HardwareProjection;

// hardwareProjection can be null in case of the actual driveObject
// does not support activation.
// Setting the required MotorType and DriveDataSetNumber on the drive.
// It is only supported on G120 drives.

//First parameter is the MotorType, Second parameter is the DriveDataSetNumber
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

// There can be a problem, if the selected MotorType is not available
// on the drive. In that case, it has to give back a feedback.
```

Setting the motor type online via the hardware configuration

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

//Online (Only on G120 drives)
OnlineDriveFunctionInterface onlineDfi = ...
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// hardwareProjection can be null in case of the actual
// onlineDriveObject does not support activation.
// Setting the required MotorType and DriveDataSetNumber on
// the drive. It is only supported on G120 drives.

// First parameter is the MotorType, Second parameter is the DriveDataSetNumber
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

// There can be a problem, if the selected MotorType is not available on
// the drive. In that case, it has to give back a feedback.
```

The following examples show how you can read out the motor type for G120 drives from the drive:

Determining the motor configuration offline

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
// Offline
// WARNING: You have to set the MotorType on G drives before
// you would like to get the current configuration, otherwise
// you will get an exception/feedback, which informs you that
// there is no motor set.
// On S120 drives, you don't have to do that, but you have to
// plug a motor to motor modul.( later on)

HardwareProjection hardwareProjection = dfi.HardwareProjection;
// Get the current motor configuration
// It needs a datasetnumber, which is currently only supported on G120 drives.
MotorConfiguration motorConfiguration = hardwareProjection.GetCurrentMotorConfiguration(0);
```

Determining the motor configuration online

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
// Online
// WARNING: You have to set the MotorType on G drives before
// you would like to get the current configuration, otherwise
// you will get an exception/feedback, which informs you that
// there is no motor set.

HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;
// Get the current motor configuration
// It needs a datasetnumber, which is currently only supported on G120 drives.
MotorConfiguration motorConfiguration = hardwareProjection.GetCurrentMotorConfiguration(0);
```

The following example shows how you can create a motor configuration for G120 drives:

Configuring the motor configuration for G120 drives

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Configuring the motor configuration for G120 drives

```
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();

HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

MotorConfiguration config = hardwareProjection.GetCurrentMotorConfiguration(0);

config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
  switch (ce.Number)
  {
    case 305:
        ce.Value = 20;
        break;
    case 307:
        ce.Value = 30;
        break;
    case 311:
        ce.Value = 200;
        break;
    case 304:
        ce.Value = 450;
        break;
    case 310:
        ce.Value = 50;
        break;
    case 335:
        ce.Value = 1;
        break;
    default:
        break;
  }
});
bool result =
cuDriveFunctionInterface.HardwareProjection.ProjectMotorConfiguration(config, 0);
```

The following examples show how you can configure motors for S120 and G220 drives:

Configuring motors for S120 and G220 drives

```
private bool S120ProjectMotorConfigurationTestWithName_ServoInductionMotor()
 {
 m_Device.PlugNew(@"OrderNumber:6SL3120-1TE15-0Axx//10002", "", 65535); //momo
 DeviceItem mmDeviceItem = m_Device.DeviceItems[2];
 mmDeviceItem.PlugNew(@"OrderNumber:XMxxxxx-xxxxx-xxxx//Motor data input-Induction
motors", "", 65535);
 DriveObject mmDriveObject =
mmDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
 DriveFunctionInterface mmDriveFunctionInterface =
mmDriveObject.GetService<DriveFunctionInterface>();
 HardwareProjection hwProjection = mmDriveFunctionInterface.HardwareProjection;
 bool result = false;
 try
 {
 MotorConfiguration motConfig = hwProjection.GetCurrentMotorConfiguration(0);
 if (motConfig.RequiredConfigurationEntries.Count == 0 ||
motConfig.OptionalConfigurationEntries.Count == 0)
 {
 return false;
 }
 motConfig.RequiredConfigurationEntries.ToList().ForEach(ce =>
 {
 //get all other properties of motor
 switch (ce.Name)
 {
 case "p305":
 ce.Value = 1000;
 break;
 case "p307":
 ce.Value = 500;
 break;
 case "p311":
 ce.Value = 30;
 break;
 case "p304":
 ce.Value = 10;
 break;
 case "p310":
 ce.Value = 20;
 break;
 case "p322":
 ce.Value = 22;
 break;
 case "p604":
```

Configuring motors for S120 and G220 drives

```
ce.Value = 30;
break;
case "p605":
ce.Value = 45;
break;
case "p350":
ce.Value = 64;
break;
case "p354":
ce.Value = 65;
break;
case "p356":
ce.Value = 66;
break;
case "p358":
ce.Value = 67;
break;
case "p360":
ce.Value = 68;
break;
default:
break;
}
});
motConfig.OptionalConfigurationEntries.ToList().ForEach(ce =>
{
switch (ce.Name)
{
case "p308":
ce.Value = 1;
break;
case "p338":
ce.Value = 6000;
break;
case "p348":
ce.Value = 20000;
break;
default:
break;
}
});
result = hwProjection.ProjectMotorConfiguration(motConfig, 0);
}
catch (Exception ex)
```

Configuring motors for S120 and G220 drives

```
{
Console.WriteLine(ex.Message + "\n" + ex.StackTrace);
TestInterface.Log.Error(ex.Message + "\n" + ex.StackTrace);
}
return result;
}
```

5.23.5.16 Reading and writing parameters

The following example shows how to read and write values of drive parameters. You require a drive object for access.

Access to parameters

```
using Siemens.Engineering.MC.Drives;
```

Access to parameters

```
//Access a parameter via its name
DriveParameter parameter = driveObject.Parameters.Find("p5391[0]");

//Example of reading parameter attributes
if (parameter != null)
{
  Console.WriteLine("The Name of the parameter is : " + parameter.Name);
  Console.WriteLine("The value of the parameter is : " +
parameter.Value.ToString());
  Console.WriteLine("The minvalue of the parameter is : " +
parameter.MinValue);
  Console.WriteLine("The MaxValue of the parameter is : " +
parameter.MaxValue);
  Console.WriteLine("The Unit of the parameter is : " + parameter.Unit);

  //Example for write:
  parameter.Value = 60;
}

// Access a parameter via Number and ArrayIndex// Note that
//   - arrayless parameters (e.g. p96 on G120) are indexed by -1
//   - parameters that consist of only a bit array do not count as
//     array parameters, they are indexed by -1 as well and the
//     further bit values can be accessed by the 'Bits' property
//     of the given parameter (e.g. r2139 on G120). For further
//     information about accessing bit parameters see the next
//     section of this code snippet

DriveParameter r947_6 = driveObject.Parameters.Find(947, 6); // returns
r947[6]if (r947_6 != null)
{
  Console.WriteLine("The Name of the parameter is : " + r947_6.Name);
  Console.WriteLine("The value of the parameter is : " +
r947_6.Value.ToString());
}

//Accessing bit values
DriveParameter p2720 = driveObject.Parameters.Find(2720, 0);if (p2720 !=
null)
{
  // Note that in general, pXXX.Bits[YYY] is not necessarily equivalent to
pXXX.YYY.
  // pXXX.Bits is an array of available bit values in ascending order by
their names
  // and not an array of bit values indexed by their names.
  // For example: let the available bit values be [pXXX.0, pXXX.2, pXXX.3],
then
  // pXXX.Bits[2] == pXXX.3, not pXXX.2

  DriveParameter p2720Bit1 = p2720.Bits[1]; // returns p2720[0].1
  if (p2720Bit1 != null)
  {
    Console.WriteLine("The name of the parameter is : " + p2720Bit1.Name);
    Console.WriteLine("The value of the second bit of the parameter is : "
+ p2720Bit1.Value.ToString());
  }
```

Access to parameters

```
}

//Get the enum values of a parameter
DriveParameter r47 = driveObject.Parameters.Find("r47");
foreach (var enumItem in r47.EnumValueList)
{
  Console.WriteLine("Enum value: " + enumItem.Key.ToString() + " = " +
enumItem.Value);
}
```

See also

Contacts at Siemens (https://www.automation.siemens.com/aspa_app?ci=yes&lang=en)

5.23.5.17 Reading and writing parameters online

The following example shows how to obtain a list with the available online parameters. You require an online drive object for access.

The read and write access to individual online parameters from the parameter list is identical to that in the Reading and writing parameters (Page 1140) example.

Access to online parameters

```
using Siemens.Engineering.MC.Drives;
DeviceItem item = ... //device item of the CU (e.g. :
project.Devices[0].Items[0].Items[0])
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
if (onlineDriveObject != null)
{
  var parameters = onlineDriveObject.Parameters;
}
```

5.23.5.18 Saving the parameterization

The following example shows how you can determine the parameterization from S120, S210 or G120 drives.

Determining the parameterization online from the drives

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Determining the parameterization online from the drives

```
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = onlineDfi.DriveDomainFunctions;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.

// driveDomainFunctions can be null, if the actual onlineDriveObject
// does not support it.

// For example: On G120 and S210 drives, you have to use the CU to get the
onlineDriveObject.
// Please, pay attention at S120 drives. Here you can use any module (CU, MotorModul,
lineModul) as onlineDriveObject but preferable the CU, because it will run on CU!
```

The following example shows how you can copy the parameterization for S120, S210 or G120 drives from RAM to ROM and thus store it in a power-independent manner.

Backing up from RAM to ROM

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
//In case of G120 device.
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
OnlineDriveObject cuDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

//In case of S120 and S210 device you should generally get the CU driveobject.
DeviceItem cuDeviceItem = m_Device.DeviceItems[0];OnlineDriveObject cuDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

// To use the function, you have to use the OnlineDriveFunctionInterface.
OnlineDriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = cuDriveFunctionInterface.DriveDomainFunctions;

// This function will perform a RAM to ROM copy on all device.
// It is important that the CU should be in online state before you call this function
anyway the program will give an exception.
// This call may take a few moments before be finished.
bool result = DriveDomainFunctions.PerformRAMtoROMCopyAllDriveObject();
```

5.23.5.19 How to assign a SecureString password

With the SetPassword() method you can assign passwords for secure communication during downloading. The parameter must be a SecureString.

The following example shows how to assign a SecureString password.

How to assign a SecureString password

How to assign a SecureString password

```
public void SetPassword(DownloadPasswordConfiguration
downloadPasswordConfiguration)
        {
            if (downloadPasswordConfiguration != null)
           {
                SecureString s = GetSecureString();
                downloadPasswordConfiguration.SetPassword(s);
                return;
            }
        }

  private SecureString GetSecureString()
        {
            SecureString securePwd = new SecureString();
            ConsoleKeyInfo key;

            Console.Write("Enter password: ");
            do
            {
                key = Console.ReadKey(true);

                // Ignore any key out of range.
                if (((int)key.Key) >= 65 && ((int)key.Key <= 90))
                {
                    // Append the character to the password.
                    securePwd.AppendChar(key.KeyChar);
                    Console.Write("*");
                }
                // Exit if Enter key is pressed.
            } while (key.Key != ConsoleKey.Enter);

            return securePwd;

        }
```

See also

Encrypted communication with SecureString passwords (Page 1087)

5.23.5.20 Configuring Safety Integrated (checksums) (from SINAMICS FW V6.1)

The following examples show how you can configure Safety Integrated Functions via checksums for SINAMICS drives from firmware version V6.1 with Openness. This function is only available in the offline mode.

Before accessing `UpdateCheckSums`, the drive object must first be determined (Page 1113) via `DriveObject`.

Configuring Safety Integrated via checksums (only offline)

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
```

Configuring Safety Integrated via checksums (only offline)

```
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface driveFunctionInterface =
cuDriveObject .GetService<DriveFunctionInterface>();

if (driveFunctionInterface.SafetyCommissioning == null)
 {
   return false;
 }

//UpdateCheckSums throws exception in case, if the specified CoreObject is null or not
offline
bool retValue = driveFunctionInterface.SafetyCommissioning.UpdateCheckSums();

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is online.

// G120 and S120 drives(drives with SINAMICS Firmware version older than V6.1) do not
support it.
// For G120 drives(CU250S-2_PN), driveFunctionInterface.SafetyCommissioning returns
'UpdateCheckSums is not supported for 2nd gen drives'
// For S120 acx drives, driveFunctionInterface.SafetyCommissioning returns null
```

5.23.5.21 Using Safety Integrated telegrams

The following example shows how you can use Safety Integrated telegrams (e.g. 30) in Openness.

Using Safety Integrated telegrams

```
using Siemens.Engineering.MC.Drives;
TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram(tgrmNumber);

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
uint watchDogTime = (uint)safetyTgrm.GetAttribute("Failsafe_FMonitoringtime");

safetyTgrm.SetAttribute("Failsafe_FMonitoringtime", 300);

const int newSafetyTelegramNumber= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber))
{
  safetyTgrm.TelegramNumber = newSafetyTelegramNumber;
}

//Remove Safety telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);
```

5.23.5.22 Setting the SIMOGEAR article number for G115D drives

The following example shows how you can set the SIMOGEAR article number for G115D drives.

Setting the SIMOGEAR article number for G115D drives

```
using Siemens.Engineering.MC.Drives;
Dvar commissioning= dfi.Commissioning;
var commissioning= dfi.Commissioning;
//Commissioning can be null if the actual driveobject does not support the Commissioning
(e.g. SINAMICS S devices).

//Setting the article number of the SimoGear
commissioning.SetSimoGearMlfb("2KJ8001-2EG20-4DG1-D0X");
```

See also

Commissioning (Page 1091)

5.23.5.23 Using Technology Extensions

The following examples show how you can use Technology Extensions.

Install TechnologyExtensionPackage

```
string packageFilePath = @"Y:\\TRCDATA_V1_1_0_1.tec";

TechnologyExtensionInstallationProvider tecInstallationProvider =
TiaPortal.GetService<TechnologyExtensionInstallationProvider>();
FileInfo packageFile = new FileInfo(packageFilePath);

try
{
tecInstallationProvider.Install(packageFile);
}
catch (EngineeringException ex)
{
Console.WriteLine(ex.Message + "\n" + ex.StackTrace);
}
```

Activate Technology Extension

```
TechnologyExtension tecExtension= ...
tecExtension.Activate();
return tecExtension.IsActivated;
```

Deactivate Technology Extension

```
TechnologyExtension tecExtension=...
tecExtension.Deactivate();
return tecExtension.IsActivated;
```

Uninstall TechnologyExtensionPackage

```
string tecPackageIdentifier = "TRCDATA/1100802";
bool confirmUninstallInUse = true;
TechnologyExtensionInstallationProvider tecInstallationProvider =
TiaPortal.GetService<TechnologyExtensionInstallationProvider>();
TechnologyExtensionPackage tecPackage =
tecInstallationProvider.TechnologyExtensionPackages.Find(tecPackageIdentifier);

if (tecPackage == null)
{
Console.WriteLine("the specified TechnologyExtensionPackage '{tecPackageIdentifier}' was
not found");
return;
}

try
{
tecInstallationProvider.Uninstall(tecPackage.Identifier, confirmUninstallInUse);
}
catch (EngineeringException ex)
{
Console.WriteLine(ex.Message + "\n" + ex.StackTrace);
}
```

Read out DriveObjectContainer of a SINAMICS device

```
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.MC.Drives
Device device = ...;
foreach (DeviceItem deviceItem in device.DeviceItems)
{
DriveObjectContainer doContainer = deviceItem.GetService<DriveObjectContainer>();
// do something
}
```

Navigate to a DriveObject

```
Device device = ...;
foreach (DeviceItem deviceItem in device.DeviceItems)
{
DriveObjectContainer doContainer = deviceItem.GetService<DriveObjectContainer>();
DriveObjectComposition driveObjects = doContainer.DriveObjects;
foreach (DriveObject driveObject in driveObjects)
{
// do something with the DriveObject
}
}
```

Navigate to a Technology Extension

```
DriveObject driveObject = ...
TechnologyExtensionContainer tecContainer =
driveObject.GetService<TechnologyExtensionContainer>();
TechnologyExtensionComposition tecExtensions = tecContainer.TechnologyExtensions;
foreach (TechnologyExtension tecExtension in tecExtensions)
{
// do something with the TechnologyExtensions
}
```

Find TechnologyExtension using TechnologyExtension.Name

```
DriveObject driveObject = ...;
TechnologyExtensionContainer tecContainer =
driveObject.GetService<TechnologyExtensionContainer>();
TechnologyExtension tecExtension = tecContainer.TechnologyExtensions.FirstOrDefault(x =>
x.Name == tecExtensionName);
// do something with the TechnologyExtension
```

See also

TechnologyExtension (Page 1102)

5.23.5.24 Connecting technology objects with hardware modules via telegram objects

You can connect technology objects with hardware modules via telegram objects using method Connect.

Connecting technology objects with hardware modules via telegram objects

```
private bool Test()
{
    Device plcStation = m_Project.Devices[0];
    DriveObject driveObject =
m_Project.Devices.ElementAt(1).DeviceItems[2].GetService<DriveObjectContai
ner>().DriveObjects[0];
    Telegram telegram = driveObject.Telegrams[0];

    DeviceItem plcDevice = plcStation.DeviceItems[1];
    SoftwareContainer softwareContainer =
plcDevice.GetService<SoftwareContainer>();
    Software software = softwareContainer.Software;

    TechnologicalInstanceDB techObj0 =
((Siemens.Engineering.SW.PlcSoftware)software).TechnologicalObjectGroup.Te
chnologicalObjects[0];
    AxisHardwareConnectionProvider axisHardwareConnectionProvider0 =
techObj0.GetService<AxisHardwareConnectionProvider>();
    AxisEncoderHardwareConnectionInterface
axisEncoderHardwareConnectionInterfaces0 =
axisHardwareConnectionProvider0.ActorInterface;

    try
    {
        axisEncoderHardwareConnectionInterfaces0.Connect(telegram); //
Connect API
    }
    catch (Exception e)
    {

    }
}
```

Connecting technology objects with a hardware module via a telegram object and ConnectOption.

```
private bool Test()
{
    Device plcStation = m_Project.Devices[0];
    DriveObject driveObject =
m_Project.Devices.ElementAt(1).DeviceItems[2].GetService<DriveObjectContai
ner>().DriveObjects[0];
    Telegram telegram = driveObject.Telegrams[0];

    DeviceItem plcDevice = plcStation.DeviceItems[1];
    SoftwareContainer softwareContainer =
plcDevice.GetService<SoftwareContainer>();
    Software software = softwareContainer.Software;

    TechnologicalInstanceDB techObj0 =
((Siemens.Engineering.SW.PlcSoftware)software).TechnologicalObjectGroup.Te
chnologicalObjects[0];
    AxisHardwareConnectionProvider axisHardwareConnectionProvider0 =
techObj0.GetService<AxisHardwareConnectionProvider>();
    AxisEncoderHardwareConnectionInterface
axisEncoderHardwareConnectionInterfaces0 =
axisHardwareConnectionProvider0.ActorInterface;

    try
    {
        //Connect API
        axisEncoderHardwareConnectionInterfaces0.
        Connect(telegram,
Siemens.Engineering.MC.Drives.Enums.ConnectOption.Default); //Default = 0,
AllowAllModules = 1
        //Or the following way can also be used
        axisEncoderHardwareConnectionInterfaces0.Connect(telegram, 0); //
Default = 0, AllowAllModules = 1
    }
    catch (Exception e)
    {

    }
}
```

See also

HardwareProjection (Page 1097)

5.23.5.25 Inserting and extending telegrams

The following example shows how to insert an extension and change the size of a standard telegram. You require a drive object for access.

Inserting an extension and changing the size of a standard telegram

```
using Siemens.Engineering.MC.Drives;
```

Inserting an extension and changing the size of a standard telegram

```
TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelegramNumber);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: "
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
  if (address.IoType == AddressIoType.Output)
  {
    Console.WriteLine("The Setpoint channel-specific starting address of
the connected PLC is: " + address.StartAddress);
  }
  else if(address.IoType == AddressIoType.Input)
  {
    Console.WriteLine("The Actual value channel-specific starting address
of the connected PLC is: " + address.StartAddress);
  }
}

// Add an additional Telegram
if (drvObj.Telegrams.CanInsertAdditionalTelegram(2,4))
{
  drvObj.Telegrams.InsertAdditionalTelegram(2,4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
  mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}
```

5.23.5.26 Using torque telegrams

The following example shows how you can use torque telegrams (e.g. 750) in Openness.

Using torque telegrams

```
using Siemens.Engineering.MC.Drives;
const int torqueTelegramNumber = 750;
TelegramComposition telegrams = drvObj.Telegrams;

// Add a new torque telegram
if (drvObj.Telegrams.CanInsertTorqueTelegram(torqueTelegramNumber))
{
    drvObj.Telegrams.InsertTorqueTelegram(torqueTelegramNumber);
}

// Find torque telegram
Telegram torqueTelegram = drvObj.Telegrams.Find(TelegramType.TorqueTelegram);
```

5.23.5.27 Activating/deactivating UMAC for drives

The following examples show how you can activate and deactivate function User Management and Access Control (UMAC) for SINAMICS drives from firmware version V6.1 with Openness. With Openness, UMAC can only be activated or deactivated in the offline mode.

Before accessing `UmacConfiguration`, the drive object must first be determined (Page 1113) via `DriveObject`.

Activating UMAC (only offline)

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.Security;
DeviceItem cuDeviceItem = device.DeviceItems[0];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
Security cusecurity=cuDriveObject.Security;
if(cuSecurity == null)
//For G120 drives(CU250S-2_PN), as UmacConfiguration is not supported for drives with
SINAMICS Firmware version older than V6.1, the value of ''Security'' object will be null
{
    return false;
}
UmacConfiguration umacConfiguration = cusecurity.UmacConfiguration;

//Activate the UMAC for drive
bbool result = umacConfiguration.Activate();
```

Deactivating UMAC (only offline)

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.Security;
//For online
OnlineDriveObject cuOnlineDriveObject =
device.DeviceItems[itemPosition].GetService<OnlineDriveObjectContainer>().OnlineDriveObjec
ts[0];

//Activate the DriveDataEncryption for the drive
bool result = cuOnlineDriveObject.Security.DriveDataEncryption.Activate(curpassword);

//Deactivate the DriveDataEncryption for drive
bool result = cuOnlineDriveObject.Security.DriveDataEncryption.Deactivate(oldpassword);

//Deactivate the UMAC for the drive

bool result = umacConfiguration.Deactivate();
```

5.23.5.28 Activating the encryption of drive data/DDE (from SINAMICS FW V6.1)

The following examples show how you can activate and deactivate the encryption of drive data (DDE, Drive Data Encryption) for SINAMICS drives from firmware version V6.1 with Openness.

Before accessing `DriveDataEncryption`, the drive object must first be determined (Page 1113) via `DriveObject`.

Activating and deactivating the encryption of drive data / DDE offline

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.Security;
//For offline
DriveObject driveObject =
device.DeviceItems[itemPosition].GetService<DriveObjectContainer>().DriveO
bjects[0];

//Activate the DriveDataEncryption for the drive
bool result =
driveObject.Security.DriveDataEncryption.Activate(curpassword);

//Deactivate the DriveDataEncryption for drive
bool result =
driveObject.Security.DriveDataEncryption.Deactivate(oldpassword);
```

Activating and deactivating the encryption of drive data / DDE online

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.Security;
//For online
OnlineDriveObject cuOnlineDriveObject =
device.DeviceItems[itemPosition].GetService<OnlineDriveObjectContainer>().
OnlineDriveObjects[0];

//Activate the DriveDataEncryption for the drive
bool result =
cuOnlineDriveObject.Security.DriveDataEncryption.Activate(curpassword);

//Deactivate the DriveDataEncryption for drive
bool result =
cuOnlineDriveObject.Security.DriveDataEncryption.Deactivate(oldpassword);
```

5.23.5.29 Restoring factory settings

The following example shows how you can restore the factory settings for S120, S210 or G120 drives.

Restoring factory settings

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Restoring factory settings

```
OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = onlineDfi.DriveDomainFunctions;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
// saveParametrization can be null, if the actual onlineDriveObject
// does not support it.
// For example: On G120 and S210 drives, you have to use the CU to get the
onlineDriveObject.
// S120 drives, you can use any module (CU, MotorModul, lineModul) as onlineDriveObject but
preferable the CU.

//In case of G120 device
DeviceItem cuDeviceItem = g120Device.DeviceItems[1];
OnlineDriveObject cuDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

//In case of S120 and S210 devices you should generally get the CU driveobject
DeviceItem cuDeviceItem = s120Device.DeviceItems[0]
OnlineDriveObject cuDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

// To use the function, you have to use the OnlineDriveFunctionInterface
OnlineDriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = cuDriveFunctionInterface.DriveDomainFunctions;

// This function will perform a FactoryReset for S120 drives
// It is important that the CU should be in online state before you call this function
anyway the program will give an exception.
// This call may take a few moments before be finished.
bool result = driveDomainFunctions.PerformFactoryReset(ResetMode.SafetyParameterReset);

//G120 drives have 2 flags for the ResetMode
The flag ParameterReset is used to reset the norml parameters.
The flag SafetyParameterReset is used to reset the safety parameters
```

5.24 Functions for SIMATIC Drive Controller

5.24.1 Accessing PLC and X142 channel properties

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to access:

- Parameters of the PLC
- Parameters of X142 onboard I/Os for each channel

PLC parameter

In addition to SIMATIC S7-1500 PLC attributes, the SIMATIC Drive Controller provides the following device specific attributes:

Property name	Data type	Writeable	Access
TimeSynchronizationDriveActive	bool	r/w	Dynamic attribute

The TimeSynchronizationDriveActive is a enum with following value:

Property	Description	Value
TimeSynchronizationDriveActive	Disabled	0
	Enabled	1

X142 parameter

The SIMATIC Drive Controller provides the following Drive Controller specific attributes for the X142 onboard I/Os.

X142 Basic parameter

Attribute	Data type	Writable	Description
PulseWidthModulationFrequency	UInt64	r/w	Base frequency of pulse width modulation

The PulseWidthModulationFrequency is a enum property with the following value:

Enum	Value
goverance	0
Value01Kilohertz	1
Value02Kilohertz	2
Value04Kilohertz	3
Value08Kilohertz	4
Value16Kilohertz	5

X142 channel parameter

Attribute	Data type	Writeable	Description
IoType	Siemens.Engineering.HW.ChannelIoType	r	Value: Complex Complex I/O types, e.g. for technological channels
Type	Siemens.Engineering.HW.ChannelType	r	Value: Technology The channel type is technology
Number	Int32	r	The channel number is 0 ... 7
OperatingMode	UInt64	r/w	Selection of the operating modes for the individual channels
Invert	Boolean	r/w	The signal can be inverted
InputDelay	UInt64	r/w	To suppress faults, an input delay of 1 µs or 125 µs for the input filter of the digital inputs can be set
HighSpeedOutput	Boolean	r/w	If the high-speed output option is set, the digital output is switched alternately to 24 V DC and ground. This allows extremely steep edges (output delay in the 1 µs)

The OperatingMode is a enum property with the following value:

Enum	Value
DigitalInput	52
DigitalOutput	276
TimerDQ	8
TimerDI	34
OversamplingInput	283
OversamplingOutput	284
PeriodMeasurement	31
PulseWidthModulation	7

The InputDelay is enum property with following value:

Enum	Value
Value001Microseconds	33
Value125Microseconds	34

Program code

Modify the following program code to get and set values of dynamic attributes:

```
// To create a SIMATIC Drive Controller
var driveController =
MyTiaPortal.Projects.First().Devices.CreateWithItem("OrderNumber:6ES7 615-4DF10-0AB0/
V2.8","PLC_1", "PLC_1");
// To get channels of DI/DQ 8x24VDC_1
DeviceItem di_dq = driveController.DeviceItems.First(x => x.Classification ==
Classification.CPU).DeviceItems[7];
ChannelComposition channels = di_dq.Channels;
foreach (Channel channel in channels)
{
… // work with the channel
}
// To get properties of a channel:
Channel channel = channels[0];
int channelNumber = channel.Number;
// To get values of dynamic attributes:
var isInvert = channel.GetAttribute(“Invert”);
var operatingMode = channel.GetAttribute(“OperatingMode”);

// To set values of a channel:
channel.SetAttribute(“OperatingMode”,OperatingMode.TimerDQ);
```

5.24.2 Accessing PROFIdrive Integrated properties

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to create PROFIdrive Integrated subnet and access the properties of a PROFIdrive Integrated.

Property

The following properties of PROFIdrive Integrated with ischronous can be accessed via TIA Portal Openness:

Attribute	Data type	Writable	Access	Description
Name	string	r/w	dynamic	Name of the subnet
NetType	NetType	r	modelled	Type of the subnet
IsochronousMode	bool	r	dynamic	Enabled constant bus cycle time
SourceCycleTime*	Int32	r/w	dynamic	The bus source cycle time
DpCycleTime	double	r/w	dynamic	The DP cycle time
IsochronousTiToAutoCalculation	bool	r/w	dynamic	True if automatic calculation and setting of values of Ti (read in process values) and To (output process values)
IsochronousTi	double	r/w	dynamic	Time Ti (read in process values)
IsochronousTo	double	r/w	dynamic	Time To (output process values)

The *SourceCycleTime supports the following Enums value:

Enum Name	Value
Manual	0
AutomaticMinimum	1
LocalSendClock	2
ProfinetSendClock	3

Program code

Modify the following program code to access the properties of PROFIdrive Integrated:

```
Siemens.Engineering.HW.SubnetComposition subnetComposition =
MyTiaPortal.Projects.First().Subnets;
Subnet profiDriveSubnet = subnetComposition.Create("System:Subnet.ProfidriveIntegrated",
"PROFIdrive Integrated_1");
//Couple PROFIdrive Integrated to X150 (PROFINET)
var profinetX150 = driveController.DeviceItems.First(x => x.Classification ==
Classification.CPU).DeviceItems[4].GetService<NetworkInterface>().Nodes.First();
profinetX150.ConnectToSubnet(profiDriveSubnet);
```

5.24.3 Creating SINAMICS Integrated

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to create a SINAMICS Integrated.

Program code

```
Siemens.Engineering.HW.DeviceComposition deviceComposition = project.Devices;
Device S120 = deviceComposition.CreateWithItem(@"OrderNumber:6ES7615-xDS1x-0xxx/V5.2/
S120", "Integrated_1", "SINAMICS S_1");
```

5.24.4 Getting address of central module (X142)

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to get address of X142 central module:

```
//Simatic Drive Controller is usually created in a DeviceGroup (group)
//Getting headmodule of the SIMATIC Drive Controller
DeviceItem plcHeadModule = group.Devices[0].DeviceItems[1];
//Getting address of central module (X142)
Address x142address = plcHeadModule.DeviceItems.First(x => x.PositionNumber ==
8).Addresses[0];
//Turn on isochronous mode
x142address.SetAttribute("IsochronousMode", true);
```

5.24.5 Getting address of drive controller telegram

Requirement

- The Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to get address of drive controller telegram:

```
//Simatic Drive Controller is usually created in a DeviceGroup (group)
//Getting headmodule of the S120
DeviceItem s120HeadModule = group.Devices[1].DeviceItems[0];
//Getting address of drive controller telegram
DriveObjectContainer driveObjectService =
S120DeviceItem.GetService<DriveObjectContainer>();
Telegram telegram = driveObjectService.DriveObjects.First().Telegrams.First();
Address telegramAddress = telegram.Addresses[0];
//Turn on isochronous mode
telegramAddress.SetAttribute("IsochronousMode", true);
```

5.25 Functions for DCC

5.25.1 Introduction

Using TIA Portal-Openness, you automate the engineering and control the TIA Portal using a program that you created yourself.

In this help document, you can find a lot of information and code examples for this program that you generate yourself. You can also generate and use your own programs for the TIA Portal "DCC" application.

Before you generate your own program for DCC from the sample codes listed in the following, please note the general information on Openness, which you can find in this help under the following keywords:

- Preconditions for using TIA Portal Openness
- Installing TIA Portal Openness
- Accessing the TIA Portal
- TIA Portal Openness object model
- Programming steps

5.25.2 DCC Openness

Introduction

You program applications that automate engineering in the TIA Portal using TIA Portal Openness V18.

Openness functions in conjunction with SINAMICS DCC

The following functions have been implemented in Version V18:

- Create/delete charts/subcharts/blocks
- Rename charts/blocks
- Edit comments (chart, block, pin)
- Change pin value
- Arrange blocks/subcharts
- Set blocks as predecessor
- Change run sequence of charts/blocks
- Publish/edit parameters including parameter text
- Create/delete pin-to-pin connections
- Search for blocks/subcharts
- Check DCB Extension library (whether it has been imported and where it is used)
- Import charts (Page 1180)
- Export one chart (Page 1180) or Export all charts ("Charts" folder) (Page 1173)
- Import DCB Extension libraries into the project library (Page 1174)
- Find charts based on their names (Page 1178)
- Delete a chart from the "Charts" folder (Page 1179)
- Optimize the block run sequence in a chart (Page 1173)

Note

Optimizing performance

To optimize performance when using Openness, closing the affected editors is recommended (e.g. the DCC Editor for charts, the parameter list, etc.) if an active TIA Portal instance is open with the user interface.

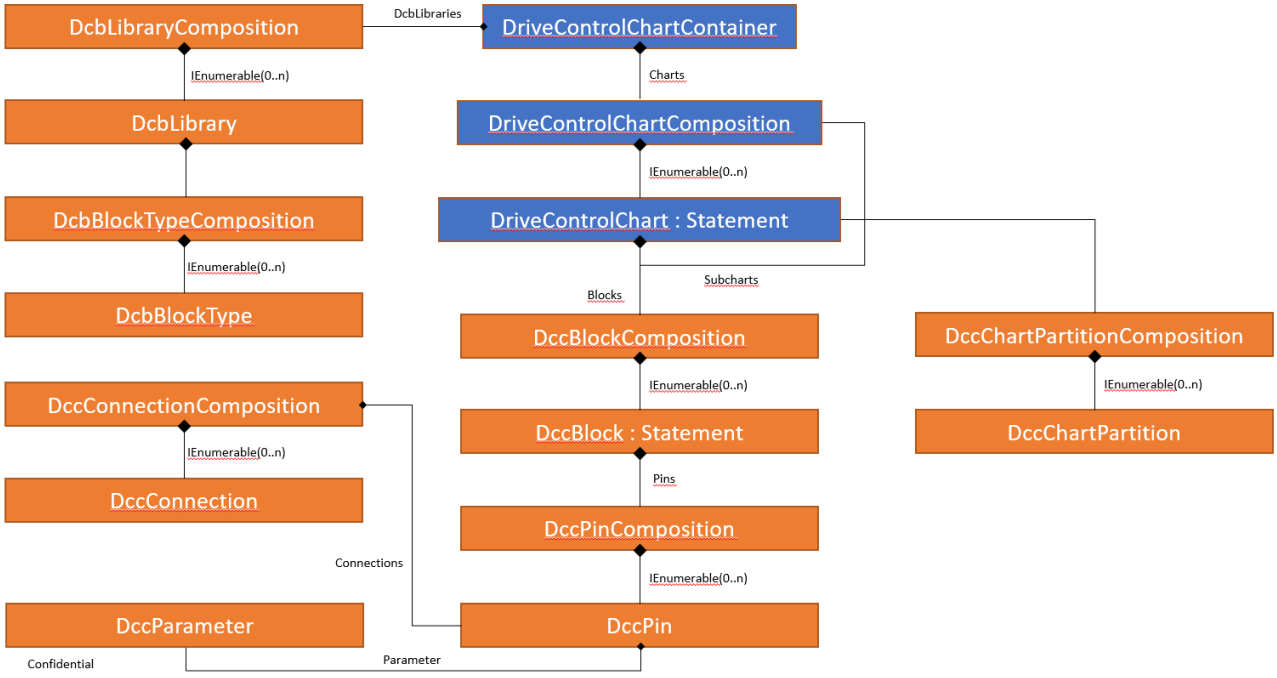
More information

See TIA information system "Openness: Automate project creation".

5.25.3 DCC Openness object model

Overview

The following diagram describes the DCC Openness object model:



5.25.4 References

5.25.4.1 DccBlock

DccBlock Features

DccBlock corresponds to a DCC block within a DCC chart.

The following table describes the **attributes** of the DccBlock:

All attributes of Statement as well as the following.

Attribute name	Data type	Writable	Description	Exceptions
GenericInputsNumber	ushort	Writable	The number of the generic input pin of the block.	DccInvalidValueException

The following table describes the **Navigators** of the DccBlock:

Name	Data type	Writable	Description
Pins	DccPinComposition	Read only	Returns all pins of a block.

The following table describes the **methods** of DccBlock:

Name	Type of return value	Parameter	Description	Exceptions
Delete	-	-	Deletes a block from a chart.	DccException

5.25.4.2 DccBlockComposition

DccBlockComposition Features

DccBlockComposition corresponds to a set of DCB blocks within a chart or subchart.

The following table describes the **methods** of DccBlockComposition:

Name	Type of return value	Parameter	Description	Exceptions
Create	DccBlock	string name string Blocktype	Creates a new DCC block in the chart with the specified name and block type name.	DccInvalidNameException, DccTextTooLongException, DccNameAlreadyUsedException, DccBlockTypeNotFoundException
Create	DccBlock	string name string Blocktype string LibraryName	Creates a new DCC block in the chart with the specified name and block type name, and uses the block type names from the selected DCB library.	DccBlockCreationInvalidNameException, DccBlockCreationNameTooLongException, DccBlockCreationBlocktypeNotFoundException
Find	DccBlock	string name	Finds a DCC block based on its name.	-

5.25.4.3 DcbLibrary

DcbLibrary

DcbLibrary corresponds to a DCB library which is used in a DCC chart container.

The following table describes the **attributes** of the DcbLibrary:

Attribute name	Data type	Writable	Description
LibraryName	string	Read only	Name of DCB library.
Version	system.Version	Read only	Version of DCB library.

The following table describes the **Navigators** of the DcbLibrary:

Name	Data type	Writable	Description
`DcbBlockTypes`	DcbBlockTypeComposition	Read only	Returns all block types that are used in the DCB library.

5.25.4.4 DcbBlockType

DcbBlockType

DcbBlockType corresponds to a DCB block type, which is available in a DCB library.

The following table describes the **attributes** of the DcbBlockType:

Attribute name	Data type	Writable	Description
`Name`	string	Read only	The name of the DCB block type.
`Description`	string	Read only	A brief description of the DCB block type.

5.25.4.5 DcbLibraryComposition

DcbLibraryComposition Features

DcbLibraryComposition corresponds to a set of DCB libraries within a certain DCC chart container.

The following table describes the **methods** of DcbLibraryComposition:

Name	Type of return value	Parameter	Description	Exceptions
`Find`	DcbLibrary-Composition	string libraryName	Finds a DcbLibrary based on its name.	-

5.25.4.6 DccConnection

DccConnection Features

DccConnection corresponds to an internal or external connection in a DCC chart, where either the sink, the source or both are a DCC object (pin or parameter).

The following table describes the **Navigators** of the DccConnection:

Name	Data type	Writable	Description
Sink	IEngineeringObject	Read only	The sink of a connection. Can be: DccPin.
Source	IEngineeringObject	Read only	The source of a connection. Can be: DccPin.

The following table describes the **methods** of DccConnection:

Name	Type of return value	Parameter	Description	Exceptions
Delete	-	-	Deletes a connection.	-

5.25.4.7 DccImportOptions

DccImportOptions

These settings are possible when importing charts:

- DccImportOptions.None:
  The import has been successfully completed if there is no name conflict. Otherwise exception DccImportChartWithSameNameAlreadyAvailableException is thrown.
- DccImportOptions.RenameOnConflict:
  If there is a chart name conflict, the CFC automatically assigns a name to the newly imported chart. For example, if a chart with the name CFC_1 already exists on import, then the newly imported chart is assigned name CFC_2.

See also

DriveControlChartComposition (Page 1170)

5.25.4.8 DccChartPartitionComposition

DccChartPartitionComposition Features

DccChartPartitionComposition corresponds to a set of partitions in DriveControlChart.

The following table describes the **methods** of DccChartPartitionComposition:

Name	Type of return value	Parameter	Description	Exceptions
Create	DccChartPartition	string name	Creates a new partition in the chart with the specified name.	DccTextTooLongException, DccNameAlreadyUsedException, DccTooManyPartitionsException
Find	DccChartPartition	string name	Finds a partition based on its name.	-

5.25.4.9 DccImportResultData

DriveParameter

DccImportResultData contains information about the result when importing charts.

The following table describes the **attributes** of DccImportResultData:

Attribute name	Data type	Writable	Description
RemappedParameterNumbers	IDictionary<UInt16,UInt16>	Read only	When importing, it is possible that a DCC parameter to be imported is already available. In this case, the imported parameter is assigned a new number. This dictionary then contains all of the newly assigned parameters. The key value is the previous parameter number as it was exported and the value is the newly created parameter.

See also

DriveControlChartComposition (Page 1170)

5.25.4.10 DccParameter

DccParameter

DccParameter corresponds to a parameter, which is published in a DCC chart at a DCC pin.

The following table describes the **attributes** of DccParameter:

Attribute name	Data type	Writable	Description	Exceptions
`Number`	ushort	Writable	The parameter number of the DCC parameter.	DccParameterNumberAlreadyExistsException, DccParameterNumberOutOfRangeException
`ParameterText`	string	Writable	The parameter text of the DCC parameter.	DccTextTooLongException
`IsSink`	boolean	Read only	Indicates whether the DCC parameter is a sink or a source.	
`IsSignal`	boolean	Writable	Indicates whether the parameter is a signal parameter.	

5.25.4.11 DccPin

DccPin Features

DccPin corresponds to a pin in a DCC block instance chart within a DCC chart.

The following table describes the **attributes** of the DccPin:

Attribute name	Data type	Writable	Description	Exception
Name	string	Read only	Name of the pin.	-
Comment	string	Writable	The comments of the pin.	TextTooLongException
Value	object	Writable	The value of the pin.	DccInvalidValueException
IsInput	boolean	Read only	True if the pin is an input, otherwise false.	-
Invisible	boolean	Writable	Indicates whether the pin is invisible or not.	-
Unit	string	Writable	The unit of the pin.	-
ForTest	boolean	Writable	Indicates whether the pin is set to monitoring or not.	-

The following table describes the **Navigators** of the DccPin:

Name	Data type	Writable	Description
`Connections`	DccConnectionComposition	Read only	All connections of the pin.
`Parameter`	DccParameter	Read only	The DCC parameters that are published with the pin.

The following table describes the **methods** of DccPin:

Name	Type of return value	Parameter	Description	Exceptions
Connect	DccConnection	DccPin partner	Creates a pin-to-pin connection with the specified partner pin	DccIllegalAssignmentException
Publish	DccParameter	bool setAsSignal	Publishes the pin as parameter. If setAsSignal is true, then the pin is published as signal, and the parameter number is automatically set.	DccPublishingStructAlarmIdException, DccNoMoreFreeParameterNumberInDriveObjectException
Publish	DccParameter	bool setAsSignal, ushort parameterNumber	Publishes the pin with the specified parameter number. The pin is published as signal if setAsSignal is true.	DccParameterNumberAlreadyExistsException, DccParameterNumberOutOfRangeException, DccPublishingStructAlarmIdException, DccNoMoreFreeParameterNumberInDriveObjectException
Unpublish	-	-	Unpublishes the pin as parameter, and existing DCC connections are deleted.	-

5.25.4.12 DccPinComposition

DccPinComposition Features

DccPinComposition corresponds to a set of pins in a DccBlock instance.

The following table describes the **methods** of DccPinComposition:

Name	Type of return value	Parameter	Description	Exceptions
`Find`	DcbPin	string name	Finds a pin based on its name.	-

5.25.4.13 DriveControlChart

DriveControlChart Features

DriveControlChart corresponds to a chart or subchart.

The following table describes the **attributes** of the DriveControlChart:

Attribute name	Data type	Writable	Description	Exception
Name	string	Read only	Name of the chart.	DccTextTooLongException, DccNameInvalidException, DccNameAlreadyUsedException
Comment	string	Writable	The comment from DriveControlChart.	DccTextTooLongException
Partition	DccChartPartition	Writable	The partition at which the DriveControlChart is located in the higher-level chart.	-
PositionX	int	Writable	The X coordinate of the subchart within the higher-level chart.	-
PositionY	int	Writable	The Y coordinate of the subchart within the higher-level chart.	-
HorizontalSheets	uint	Writable	The number of horizontal pages in the chart.	DccInvalidValueException
VerticalSheets	uint	Writable	The number of vertical pages in the chart.	DccInvalidValueException
SheetWidth	int32	Read only	The width of the sheet for the actual page size.	-
SheetHeight	int32	Read only	The height of the sheet for the actual page size.	-

The following table describes the **navigators** of the DriveControlChart:

Name	Data type	Writable	Description
`Blocks`	DccBlockComposition	Read only	Returns all blocks within a chart in an unsorted sequence.
`Subcharts`	DriveControlChartComposition	Read only	Returns all subcharts of a chart in an unsorted sequence.
`Partitions`	DccPartitionComposition	Read only	Returns every Statement (subcharts and blocks) of a chart in an unsorted sequence.

The following table describes the **methods** of DriveControlChart:

Name	Type of return value	Parameter	Description	Exceptions
`Export`	-	string path	**Method description:** Exports only this chart. **Parameter description:** path: Complete path of the file to be exported.	DccExportException (Page 1185)
Find	Statement	string name bool deepSearch	Finds a subchart or block within the present chart or in the lower-level subchart if deepSearch is set to true.	-
`Delete`	-	-	**Method description:** Deletes the chart.	-
GetRunSequence	IList<Statement>	-	Returns the run sequence of the Statement within the current chart.	-
`OptimizeRunSequence`	-	-	**Description method:** Optimizes the run sequence of the chart. CFC provides the optimization mechanism.	-
ShowDccEditor	-	-	Opens the DCC Editor in the TIA Portal for the chart when in the interactive mode	DccException, DccChartIsProtectedException

See also

DriveControlChart Exceptions (Page 1185)

DriveControlChartComposition (Page 1170)

Deleting a chart (Page 1179)

5.25.4.14 DriveControlChartContainer

DriveControlChartContainer

DriveControlChartContainer is a service that represents the DCC chart container under a DriveObject, so that it can be retrieved from the appropriate DriveObject. The service is not provided if the DriveObject is not from a supported device.

The following table describes the **Navigators** of the DriveControlChartContainer:

Name	Data type	Writable	Description
`Charts`	DriveControlChartComposition	Read only	Reads out the chart layout (composition) in the chart container.
`DcbLibraries`	DcbLibraryComposition	Read only	Returns all DCB libraries that are used in the charts.

5.25.4.15 DriveControlChartComposition

DriveControlChartComposition features

DriveControlChartComposition lists the available charts.

The following table describes the **Methods** of DriveControlChartComposition:

Name	Type of return value	Parameter	Description	Exceptions
`Create`	DriveControlChart (Page 1168)	string name	Creates a new chart with the specified name.	DccTextTooLongException, DccNameInvalidException, DccNameAlreadyUsedException, DccTooManyChartsException, DccNestingTooDeepException, DccException
`Create`	DriveControlChart (Page 1168)	-	Creates a new chart with an automatically generated name.	DccTooManyChartsException, DccNestingTooDeepException, DccException
`Import`	DccImportResultData (Page 1166)	string path, DccImportOptions (Page 1165) importOptions	**Method description:** Import charts from a DCC export file. **Parameter description:** path: Complete path of the file to be imported. importOptions: Options used when importing, see DccImportOptions (Page 1165).	DccImportException (Page 1185)
Export	-	string path	**Method description:** Exports all charts to a DCC export file. **Parameter description:** path: Complete path of the file to be exported.	DccExportException (Page 1185)
GetChartSequence	DriveControlChart (Page 1168)	-	Reads-out all charts in the container in the sequence in which they are run.	-
Find	DriveControlChart (Page 1168)	string name	Identifies a chart based on its name in the chart container.	-

See also

DriveControlChartComposition exceptions (Page 1185)

DccImportResultData (Page 1166)

DriveControlChart (Page 1168)

DccImportOptions (Page 1165)

5.25.4.16 Importing DCB extension library

ImportDCBextensionlibrary

The following table describes the **Methods** of ImportDCBextensionlibrary:

Name	Return value	Parameter	Description
ImportDcbLibrary	-	string path	**Method arguments:** path: The complete path of the DCB Extension library zip file. **Method description:** Imports a DCB Extension library into the project library.

5.25.4.17 Statement

Statement

A Statement represents a DriveControlChart or a DCC block. All of the following attributes and methods are inherited.

The following table describes the **attributes** of the statement:

Attribute name	Data type	Writable	Description	Exception
Name	string	Writable	The name of the statement.	DccNameTooLongException DccNameInvalidException
Comment	string	Writable	The comment of the statement.	DccNameTooLongException DccNameInvalidException
Partition	DccChartPartition	Writable	The partition in which the statement is located in the higher-level chart.	-
PositionX	int	Writable	The X coordinate of the statement within the higher-level chart.	-
PositionY	int	Writable	The Y coordinate of the statement within the higher-level chart.	-

Note

Invalid coordinates for a statement - CFC positions automatically

If one of the coordinates (PositionX, PositionY) for the statement receives an invalid value (e.g. PositionY = -20), then CFC will automatically position the statement. As a consequence, the value of the corresponding attribute (e.g. PositionY) still continues to display the invalid value. This response is intended. This allows you to identify as to whether a statement was automatically positioned.

The following table describes the **methods** of the statement:

Name	Return value	Parameter	Description	Exceptions
`SetAsPredecessor`	-	string path	Sets the statement as the predecessor of the next statement to be added to the chart. The method has no effect when used on a source chart.	-
`MoveInRuntimeSequence`	-	int index	Moves the statement in the run sequence to the specified index.	DccInvalidValueException

5.25.5 Code examples

5.25.5.1 General

The following code examples describe the basic procedure for various applications. The code is not necessarily complete or compilable.

Note

Optimizing performance

To optimize performance when using Openness, closing the affected editors is recommended (e.g. the DCC Editor for charts, the parameter list, etc.) if an active TIA Portal instance is open with the user interface.

5.25.5.2 Changing the run sequence

The following example shows how you can change the sequence in which charts are run.

Changing the run sequence of the charts

```
DriveControlChart chart;
IList<IStatement> runtimeSequence = chart.GetRunSequence();
runtimeSequence[3].MoveInRuntimeSequence(0);
```

5.25.5.3 Optimizing the execution sequence

The following example shows how you can optimize the sequence in which charts are run.

Optimizing the run sequence of charts

```
try
{
  DriveControlChart chart;
  ...
  chart.OptimizeRunSequence();
}
catch (DccLicenseUnavailableException exc)
{
}
```

See also

DCC Openness (Page 1161)

5.25.5.4 Accessing charts

The following example shows how you can access charts.

Accessing charts

```
DriveControlChartComposition charts = ...
DriveControlChart firstChart = charts[0];

// or looping...
foreach(DriveControlChart chart in charts)
{
    ...
}
```

5.25.5.5 Export all charts

The following example shows how you can export all charts.

Exporting all charts

```
try
{
  charts.Export(@"c:\Charts.dcc");
}
catch (DccExportException exc)
{
}
```

See also

DCC Openness (Page 1161)

5.25.5.6 Importing a DCB Extension library

The following example shows how you can import a DCB Extension library.

Importing a DCB Extension library

```
try
{
  Project myProject;
  ...

myProject.ProjectLibrary.ImportDcbLibrary(@"c:\GMCV5_1_sinamics5_1_(5.1.15
).zip");
}
catch (DccImportException exc)
{
}
```

See also

DCC Openness (Page 1161)

5.25.5.7 Reading out DCB library information

The following example shows how you can read out information of a DCB library.

Reading out DCB library information

```
DriveControlChartContainer chartContainer = ...
DcbLibraryComposition dcbLibraries = chartContainer.DcbLibraries;
DcbLibraryComposition testLib = dcbLibraries.Find("TestLib");
System.Version libVersion = testLib[0].Version;
System.Console.WriteLine($"Major version: {version.Major}, Minor version:
{version.Minor});
```

5.25.5.8 Editing DCC blocks

The following example shows how you can create a block in the chart.

Creating a block in the chart

```
DriveControlChart chart;
var blocks = chart.Blocks;
DccBlock newBlock = blocks.Create("ADD")
DccBlock newBlock = blocks.Create("add_1", "ADD");
DccBlock newBlock = blocks.Create("add_1", "gear", "GMC");
```

The following example shows how you can place a block in the lower right-hand corner of the chart.

Placing a block in the lower right-hand corner of the chart

```
DriveControlChartContainer chartContainer = ...
DriveControlChart chart = chartContainer.Charts.Find("DCC_1");
DccBlockComposition blocks = chart.Blocks;
DccBlock block = blocks.Create("ADD_1");

int horizontalSheetCount = chart.HorizontalSheets;
int verticalSheetCount = chart.VerticalSheets;
int blockSizeX = 20;
int blockSizeY = 20;


int sheetWidth = chart.SheetWidth;
int sheetHeight = chart.SheetHeight;

block.PositionX = (sheetWidth * horizontalSheetCount) - blockSizeX;
block.PositionY = (sheetHeight * verticalSheetCount) - blockSizeY;
```

The following example shows how you can find a block based on its name.

Finding a block based on its name

```
DriveControlChart chart;
var blocks = chart.Blocks;
DccBlock addBlock = blocks.Find("add_1");
```

The following example shows how you can set the attributes of a block.

Setting the attributes of a block

```
DccBlock dccBlock;
dccBlock.Name = "add_3";
dccBlock.Comment = "new comment";
dccBlock.PositionX = 110;
dccBlock.PositionY = 234;
dccBlock.GenericInputsNumber = 4;
```

The following example shows how you can delete a block from a chart.

Deleting a block from a chart

```
DccBlock dccBlock = chart.Blocks.Find("add_2");
dccBlock.Delete();
```

The following example shows how you can set a statement (e.g. DCC block) as predecessor in the run sequence.

Setting a statement (e.g. DCC block) as predecessor in the run sequence.

```
DccBlock dccBlock;
dccBlock.SetAsPredecessor();
```

5.25.5.9 Reading out DCB block type information

The following example shows how you can read out information of a DCB block type.

Reading out DCB block type information

```
DriveControlChartContainer chartContainer = ...
DcbLibraryComposition dcbLibraries = chartContainer.DcbLibraries;
DcbLibraryComposition testLib = dcbLibraries.Find("TestLib");
DcbBlockTypeComposition blockTypes = testlib.DcbBlockTypes;
DcbBlockType blockType = blockTypes.Find("test_all_pin_types");

System.Console.WriteLine($"Name: {blockType.Name}, Description:
{blockType.Description});
```

5.25.5.10 Editing DCC parameters

The following example shows how you can edit DCC parameters.

Changing DCC parameters

```
DccPin pinOnControlUnit;
DccParameter param1 = pinOnControlUnit.Parameter;
param1.Number = 21600;
param1.ParameterText = "new text for parameter";
```

5.25.5.11 Displaying an import report

The following example shows how you can display a report associated with the import.

Displaying an import report

```
DccImportResultData importResultData = charts.Import(@"d:\Charts.dcc",
DccImportOptions.None);
IDictionary<ushort, ushort> remappedParameters = importResultData.RemappedP
arameterNumbers;
foreach (KeyValuePair<ushort, ushort> remappedParameter in remappedParamete
rs)
{
  System.Console.WriteLine($"ParameterNumber <{remappedParameter.Key}> has
been changed to <{remappedParameter.Value}> during import.");
}
```

5.25.5.12 Editing pins

The following example show how you can edit pins.

Finding a pin based on its name

```
DccBlock block;
DccPinComposition pins = block.Pins;
DccPin x3 = pins.Find("X3");
```

Setting the attributes of a pin

```
DccPin dccPin;
dccPin.Comment = "new comment";
dccPin.Value = 42;
```

Editing pins

```
DccBlock block1;
DccPin input = block.Pins.Find("X1");
DccPin output = block.Pins.Find("Y");
input.Connect(output);
```

Publishing pins as parameter

```
DccBlock block1;
DccPin input = block1.Pins.Find("X1");
DccPin output = block1.Pins.Find("Y");
input.Publish(true);
output.Publish(false, 21620);
```

Unpublishing pins as parameter

```
output.Unpublish();
```

5.25.5.13 Retrieving charts in the order in which they are executed

The following example shows how you can retrieve charts in the order in which they are executed.

Retrieving charts

```
DriveControlChartComposition charts = ...
IList<DriveControlChart> chartSequence = charts.GetChartSequence();
```

5.25.5.14 Finding charts or blocks using the name

The following example shows how you can search for charts or blocks using the name. There is no difference between upper and lower case when searching for charts or blocks.

Finding a chart or block

```
DriveControlChartComposition charts = ...
DriveControlChart chart = charts.Find("CFC_1");
chart.Blocks.Find("add_1");
```

See also

DCC Openness (Page 1161)

5.25.5.15 Creating a chart or subchart

The following example shows how you can create a new chart or subchart.

Creating a DCC chart or subchart

```
DriveControlChartComposition charts = ...
DriveControlChart chart1 = charts.Create("DCC_1");
DriveControlChart chart2 = charts.Create(); // the next available number
will be used, so here it will be "DCC_2"
```

5.25.5.16 Editing charts

The following examples show how you can edit charts.

Setting the attributes of a chart

```
DriveControlChart dccChart;
dccChart.Name = "DCC_3";
dccChart.Comment = "new comment";

// setting layout
dccChart.HorizontalSheets = 4;
dccChart.VerticalSheets = 3;


// applicable for a subchart
dccChart.Partition = parentChart.Partitions[1];
dccChart.PositionX = 110;
dccChart.PositionY = 234;
```

Reading out and changing the run sequence

Reading out and changing the run sequence

```
DriveControlChartchart;
IList<IStatement>runtimeSequence=chart.GetRunSequence();
runtimeSequence[3].MoveInRuntimeSequence(1);
```

Setting a chart as predecessor

```
DriveControlChart chart;
...
chart.SetAsPredecessor(); // if the chart is a root chart where the method
is called, then there is no effect as root chart cannot be predecessor in
the runtime sequence
```

Opening the DCC Editor when the TIA Portal is in the interactive mode

```
DriveControlChart chart;
...
chart.ShowDccEditor();
```

Opening the DCC Editor when the TIA Portal is in the interactive mode

```
DriveControlChart chart;
...
chart.ShowDccEditor();
```

Setting partitions

```
DriveControlChartchart;
...DccChartPartitionCompositiitonpartitions=chart.Partitions;
DccChartPartitionpartition2=partitions.Create("Partition_2");

// moving a block to the new partition
chart.Blocks.Find("add_2").Partition=parition2;
```

5.25.5.17 Deleting a chart

The following example shows how you can delete charts.

Deleting charts

Deleting charts

```
try
{
  DriveControlChart chart;
  ...
  chart.Delete();
}
{
}
```

See also

DCC Openness (Page 1161)

DriveControlChart (Page 1168)

5.25.5.18 Retrieving charts

The following example shows how you can retrieve charts.

Retrieving charts

```
DriveControlChartComposition charts = chartContainer.Charts;
```

5.25.5.19 Importing charts

The following example shows how you can import charts.

Importing charts

```
try
{
  DriveControlChartComposition charts = ...
  DccImportResultData result = charts.Import(@"c:\Charts.dcc",
DccImportOptions.None);
}
catch (DccImportException exc)
{
}
```

See also

DCC Openness (Page 1161)

5.25.5.20 Exporting charts

The following example shows how you can export charts.

Exporting charts

Exporting charts

```
try
{
  DriveControlChart chart;
  ...
  chart.Export(@"c:\CFC_1.dcc");
}
catch (DccExportException exc)
{
}
```

See also

DCC Openness (Page 1161)

5.25.5.21 Editing connections

The following example shows how you can read out the sink/source of a connection.

Reading out to the sink/source of a connection

```
DccConnection connection;
var sink = connection.Sink;
if (sink is DccParameter)
{
  System.Console.WriteLine("Sink is DCC parameter: " +
((DccParameter)sink).ParameterNumber);
}
```

5.25.5.22 Accessing a DriveControlChartContainer via DriveObject

The following example shows how you can access a DriveControlChartContainer via a drive object.

Accessing a DriveControlChartContainer

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DCC;
Project project =  portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
DriveObject
driveObject = item.GetService<DriveObjectContainer>().DriveObjects[0];
DriveControlChartContainer
chartContainer = driveObject.GetService<DriveControlChartContainer>();
//chartContainer can be null in case the DriveObject does not support DCC.
```

5.25.6 DCC Openness exceptions

5.25.6.1 Exception handling

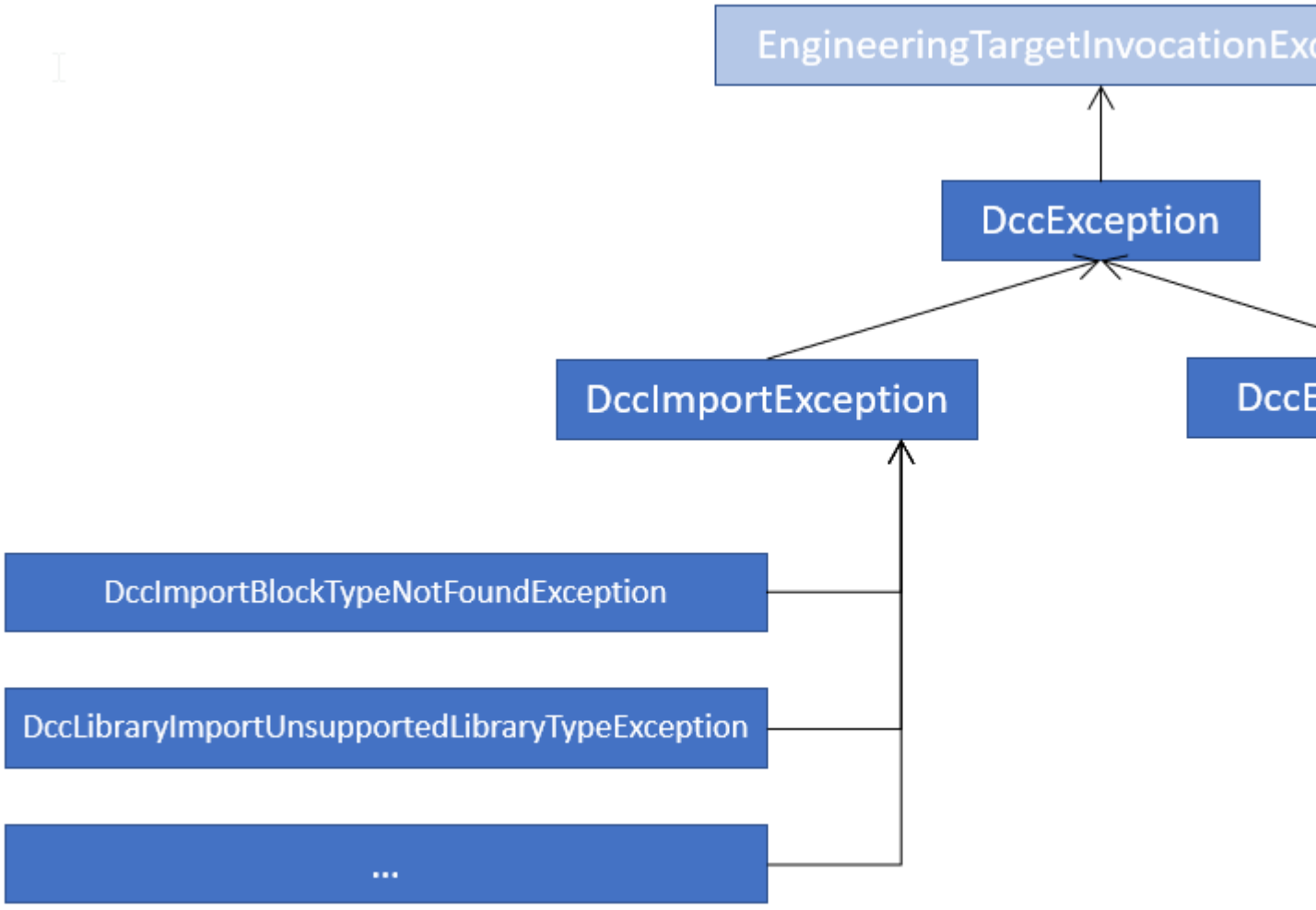
Exception handling

For applications that could fail because of user error, DCC Openness throws a DCC specific exception derived from `EngineeringTargetInvocationException`. These exceptions are hierarchical, which means that all specific exceptions are derived from the higher-level exception `DccException`:

Note

Exception if license is missing

The exception `DccLicenseUnavailableException` is thrown if the DCC license required for a function is not available.



If you do not want to investigate or evaluate different error causes in detail, then the simplest way is to intercept the general `DccException`:

```
try
{
  Project myProject;
  …

myProject.ProjectLibrary.ImportDcbLibrary(@"c:\GMCV5_1_sinamics5_1_(5.1.15
).zip");
  ...
  DriveControlChartContainer chartContainer = driveObject.GetService<DriveC
ontrolChartContainer>();
  chartContainer.Charts.Import(@"d:\Charts.dcc", DccImportOptions.None);
}
catch (DccException exc)
{
}
```

If you wish to respond based on the type of error, then catch all relevant exceptions:

```
try
{
  DriveControlChartContainer
chartContainer = driveObject.GetService<DriveControlChartContainer>();
  chartContainer.Charts.Import(@"d:\Charts.dcc", DccImportOptions.None);
}
catch (DccImportLibraryIsMissingException missingLibExc)
{
}
catch (DccImportChartWithSameNameAlreadyAvailableException sameNameExc)
{
}
catch (DccImportException exc)
{
}
catch (DccException exc)
{
}
```

5.25.6.2 DccBlockComposition Exceptions

Exceptions

Exceptions that can be thrown when creating a block:

- DccInvalidNameException:
  Is thrown if the block name is invalid.
- DccTooLongTextException:
  Is thrown if the block name is too long.

- DccNameAlreadyUsedException:
  Is thrown if the block name is already being used in the chart.
- DccBlocktypeNotFoundException:
  Is thrown if the specified block type cannot be found in any of the existing libraries.

5.25.6.3 DriveControlChart Exceptions

Exceptions

Exceptions that can be thrown when exporting:

- DccExportException:
  Is thrown for all export error types.

Exceptions that can be thrown when changing the run sequence:

- DccException:
  Is thrown if the run sequence cannot be changed.

Exceptions that can be thrown when executing the function to optimize the sequence in which charts are run.

- DccLicenseUnavailableException:
  Is thrown if a DCC license is not available. In this case, the function to optimize the run sequence is not available.

See also

DriveControlChart (Page 1168)

5.25.6.4 DriveControlChartComposition exceptions

Exceptions

Exceptions that can be thrown when importing:

- DccException:
  Is thrown if a new chart cannot be created.
  Is thrown if the run sequence of charts cannot be changed.
- DccTextTooLongException:
  Is thrown if a new chart cannot be created as the name is too long.
- DccNameInvalidException:
  Is thrown if a new chart cannot be created as the name is invalid (e.g. as the name contains invalid characters).
- DccNameAlreadyUsedException:

- DccTooManyChartsException:
  Is thrown if the maximum number of charts has been exceeded.
- DccNestingTooDeepException:
  Is thrown if the maximum number of nested subcharts has been exceeded.
- DccImportException:
  General import exception, which DCC throws in the case of an import error.
- DccImportBlockCreationException:
  Is thrown if a block was not able to be created when being imported.
- DccImportBlockTypeNotFoundException:
  Is thrown if a referenced DCB block type was not able to be found in a standard DCBLib or in an extension library.
- DccImportChartCreationException:
  Is thrown if a chart cannot be created during an import.
- DccImportChartWithSameNameAlreadyAvailableException:
  Is thrown if a chart with the same name already exists in the chart container and DccImportOptions.None is used.
- DccImportLibraryIsMissingException:
  Is thrown if a referenced DCB Extension library does not exist in the project.
- DccImportFileAlreadyInUseException:
  Is thrown if the file being imported is already being used in another process.
- DccImportDcbTypeDifferentVersionAlreadyUsedException:
  Is thrown if a DCB type in a different version of the same DCB Extension library is already being used in the device.

See also

DriveControlChartComposition (Page 1170)

5.25.6.5 ImportDcbLibrary Exceptions

Exceptions

Exceptions thrown by ImportDcbLibrary:

- DccImportException:
  Is thrown for all import error types.
- DccLibraryImportAlreadyAvailableException:
  Is thrown if the same version of the DCB Extension library already exists in the project.
- DccLibraryImportCorruptedStudioLibraryException:
  Is thrown if the DCB Extension library zip file is corrupted and cannot be imported.
- DccLibraryImportIntegrityBrokenException:
  Is thrown if the zip file of the DCB Extension library is corrupted.

- DccLibraryImportOverallPinLimitExceededException:
  Is thrown if the DCB Extension library contains a DCB type that exceeds the maximum number of pins permitted.
- DccLibraryImportStandardLibraryAlreadyAvailableException:
  Is thrown if the DCB Extension library selected for import is the standard DCBLib, and already exists.
- DccLibraryImportUnsupportedLibraryTypeException:
  Is thrown if the selected DCB Extension library is not supported by DCC and SINAMICS drives.

5.26 Exceptions

5.26.1 Handling exceptions

Exceptions when accessing the TIA Portal via TIA Portal Openness APIs

During the execution of an TIA Portal Openness application via the TIA Portal Openness API, all errors which occur are reported as exceptions. These exceptions contain information that will help you to correct the errors which have occurred.

A distinction is made between two types of exceptions:

- Recoverable (Siemens.Engineering.EngineeringException)
  You can continue to access the TIA Portal without interruption with this exception. Alternatively, you can cancel the connection to the TIA Portal.
  The EngineeringExceptions include the following types:
  - Security-related exceptions (EngineeringSecurityException), for example, in case of missing access rights.
  - Exceptions when accessing objects (EngineeringObjectDisposedException), for example, when accessing objects which no longer exist.
  - Exceptions when accessing attributes (EngineeringNotSupportedException), for example, when accessing attributes which do not exist.
  - General exceptions when calling (EngineeringTargetInvocationException), for example, error despite valid call of TIA Portal Openness API.
  - Exceptions when calling (EngineeringRuntimeException), for example, invalid cast exception.
  - Exceptions when there are not enough resources in associated TIA Portal instance (EngineeringOutOfMemoryException)
  - Exceptions when calling is terminated (EngineeringUserAbortException), for example, during an import action canceled by user.
  - Exceptions thrown during the API call invocated from a client provided delegate (EngineeringDelegateInvocationException). This exception is derived from EngineeringTargetInvocationException exception.

  The EngineeringExceptions have the following attributes:
  - `ExceptionMessageData messageData`: Contains the reason for which the exception was thrown.
  - `ExceptionMessageData detailMessageData`: Contains additional information about the reason. The result is returned as a <IList> .
  - `String message`: Returns the result from `MessageData` and `DetailMessageData`.

  `ExceptionMessageData` returns the following information:
  - `String Text`: Contains the reason for which the exception was thrown.
- NonRecoverable (Siemens.Engineering.NonRecoverableException)
  This exception closes the TIA Portal and the connection to the TIA Portal is disconnected. You need to restart the TIA Portal using the TIA Portal Openness application.

Program code

The following example shows the options you have for responding to exceptions:

```
try
{
    ...
}

catch(EngineeringSecurityException engineeringSecurityException)
{
    Console.WriteLine(engineeringSecurityException);
}

catch(EngineeringObjectDisposedException engineeringObjectDisposedException)
{
    Console.WriteLine(engineeringObjectDisposedException.Message);
}

catch(EngineeringNotSupportedException engineeringNotSupportedException)
{
    Console.WriteLine(engineeringNotSupportedException.MessageData.Text);
    Console.WriteLine();
    foreach(ExceptionMessageData detailMessageData in
engineeringNotSupportedException.DetailMessageData)
    {
        Console.WriteLine(detailMessageData.Text);
    }
}

catch (EngineeringTargetInvocationException)
{
    throw;
}

catch (EngineeringException)
{
    //Do not catch general exceptions
    throw;
}

catch(NonRecoverableException nonRecoverableException)
{
    Console.WriteLine(nonRecoverableException.Message);
}
```

5.27 F-related Openness

5.27.1 F-related Openness

Requirement

Making changes in the F-Program is only allowed in the Offline mode. In other cases an exception is thrown.

If an F-Program password has been set, the user needs to be logged in, in order to make changes in the F-Program. Otherwise, the user is reminded about the need to login to the F-Program beforehand with an exception.

The TIA Portal Openness application is connected to TIA Portal.

See "Connecting to the TIA Portal (Page 82)".

Openness service

The Openness interface (Siemens.Engineering.dll) is extended with the `GlobalSettings` service (see name area `Siemens.Engineering.Safety`) which provides two actions:

- `SafetyModificationsPossible(bool safetyModificationsPossible)`
- `UsernameForFChangeHistory(string userName)`

The Openness interface (Siemens.Engineering.dll) is extended with the `SafetySignatureProvider` service which provides three actions:

- `SafetySignatureComposition Signatures`
- `UInt64 SafetySignature.Value`
- `SafetySignatureType SafetySignature.Type`

The Openness interface (Siemens.Engineering.dll) is extended by the service `SafetyAdministration` with the following actions and properties in the namespace `Siemens.Engineering.Safety`.

- `bool IsSafetyOfflineProgramPasswordSet`
- `void SetSafetyOfflineProgramPassword(SecureString newPassword)`
- `void RevokeSafetyOfflineProgramPassword(SecureString currentPassword)`
- `bool IsLoggedOnToSafetyOfflineProgram`
- `void LoginToSafetyOfflineProgram(SecureString currentPassword)`
- `void LogoffFromSafetyOfflineProgram()`

The Openness interface (Siemens.Engineering.dll) is extended by the service `SafetyAdministration`, which is accessible from a Plc-DeviceItem. The `SafetyAdministration` provides the following properties defined in the namespace

Siemens.Engineering.Safety. This two properties contain further properties that are listed in the following section.

- `RuntimegroupComposition RuntimeGroups { get; }`
- `SafetySettings Settings { get; }`

The Openness interface (Siemens.Engineering.dll) is extended by the service `SafetyPrintout` with the following action in the namespace `Siemens.Engineering.Safety`:

- `bool Print(SafetyPrintoutFilePrinter filePrinter, FileInfo fullOutputPath, string documentLayout, SafetyPrintoutOption documentationOption);`

Principle

Get the `Safety.GlobalSettings` service from the `TiaPortal` instance:
`Engineering.Safety.GlobalSettings globalSettings = TiaPortal.GetService<Engineering.Safety.GlobalSettings>();`

5.27.2 SafetyModificationsPossible

Application

The `SafetyModificationsPossible(bool safetyModificationsPossible)` action of the `GlobalSettings` service is used to prevent changes to the safety program in TIA Portal.

When the parameter `safetyModificationsPossible` is `true`, TIA Portal behaves according to the current safety settings for the safety program.

If the parameter `safetyModificationsPossible` is `false`, all changes to the safety program are locked, regardless of whether the user has already entered the password for changing the safety program or not. The system uses feedback messages to provide information that the current user is not authorized to change the safety program.

If no password has been configured or assigned for the safety program, `safetyModificationsPossible` false has no effect. This means the safety program can be changed. However, the setting up of new passwords is blocked.

The table below shows the parameters required by the method:

Name	Type	Description
`safetyModificationsPossible`	`bool`	Authorization of changes to the safety program

Program code

Prevent changes to the safety program:

```
globalSettings.SafetyModificationsPossible(false);
```

5.27.3 UsernameForFChangeHistory

Application

The action `UsernameForFChangeHistory(string userName)` specifies the user name that is used by TIA Portal for subsequent logging in the F-change history.

The maximum length of the string is limited to 256 characters. Strings which exceed the maximum length are cut off.

An empty user name (zero or empty string) resets the user name to the default value.

The table below shows the parameters required by the method:

Name	Type	Description
`userName`	string	Preferred user name

Program code

Sets the preferred user name:

```
globalSettings.UsernameForFChangeHistory("username");
```

5.27.4 SafetySignatureProvider

The signatures read out give a first indication about changes to F-blocks. For plant acceptance, continue as described in the section entitled "System acceptance" in the "SIMATIC Safety - Configuring and Programming" manual.

Application

`Signatures: SafetySignatureComposition` returns a collection of `SafetySignature` objects. Objects of this type have a property of `SafetySignature.Value: UInt64`, which return an integer value for the F-signature. In addition, the property `SafetySignature.Type: SafetySignatureType`, which returns the type of the F-signature, encapsulated in the `SafetySignature` instance.

If the F-signature does not exist or is no longer valid due to a recent change, 0 is returned. This also applies to blocks without an F-signature (e.g. system blocks).

Program code

Get the `SafetySignatureProvider` service from the `PlcBlock` instance.

```
SafetySignatureProvider signatureProvider =
block.GetService<SafetySignatureProvider>()
```

If the block is not an F-block or not a block of an F-CPU S7-1200/1500, the returned `signatureProvider` equals to null.

If there is a non-nullable block, you can query the F-signature with the following code:

```
SafetySignature signature =
signatureProvider.Signatures.Find(SafetySignatureType.BlockOfflineSi
gnature);
UInt64 value = signature.value;
```

5.27.5 SafetyAdministration

5.27.5.1 SafetyAdministration

This features are available for S7-1200/1500 F-CPUs.

A valid Safety license must be installed for actions and setting values. A missing license will result in an exception. Reading values is possible without a license.

Program code

Obtain the `SafetyAdministration` service from the PLC `DeviceItem` instance at hand.

```
SafetyAdministration safetyAdministration =
deviceItem.GetService<SafetyAdministration>();
// check if the service is not null
if (safetyAdministration != null)
{
    // ...
}
```

5.27.5.2 IsSafetyOfflineProgramPasswordSet

Program code

```
bool isPasswordSet =
safetyAdministration.IsSafetyOfflineProgramPasswordSet
```

Application

The property `IsSafetyOfflineProgramPasswordSet` returns true if the safety program password is set.

5.27.5.3 SetSafetyOfflineProgramPassword

Program code

```
if (!safetyAdministration.IsSafetyOfflineProgramPasswordSet)
{
    SecureString newPassword = new NetworkCredential("", "new
password").SecurePassword;
```

```
safetyAdministration.SetSafetyOfflineProgramPassword(newPassword);
}
```

Application

The action `SetSafetyOfflineProgramPassword` sets the safety program password if it isn't set before. After the password is set successfully, the user is not logged into the safety program. The `LoginToSafetyOfflineProgram` action needs to be called explicitly to perform a login.

The user gets feedback in the form of an exception in the following cases.

- The given password is invalid (e.g. length equals 0 or is greater than 30)
- The given password is null
- A safety program password is already set
- If the safety program cannot be changed (see "SafetyModificationsPossible (Page 1191)"), or the project is UMAC protected, manipulating the safety program password via Openness is not allowed.
- The user is not offline
- Safety license is not present

5.27.5.4 RevokeSafetyOfflineProgramPassword

Application

The action `RevokeSafetyOfflineProgramPassword` clears the F-Program password if it's correct.

After the password was revoked successfully, the user can set a valid new F-Program password with the action `SetSafetyOfflineProgramPassword`.

The user gets feedback in the form of an exception in the following cases.

- If no password is set, it cannot be revoked
- The given password is null
- If the safety program cannot be changed (see "SafetyModificationsPossible (Page 1191)"), or the project is UMAC protected, manipulating the safety program password via Openness is not allowed.
- The user is not offline
- Safety license is not present

Program code

```
if (safetyAdministration.IsSafetyOfflineProgramPasswordSet)
{
SecureString password = new NetworkCredential("", "password").SecurePassword;
safetyAdministration.RevokeSafetyOfflineProgramPassword(password);
}
```

5.27.5.5 IsLoggedOnToSafetyOfflineProgram

Program code

```
bool isLoggedOn =
safetyAdministration.IsLoggedOnToSafetyOfflineProgram
```

Application

The property `IsLoggedOnToSafetyOfflineProgram` returns true if a password is set and the user is logged on.

5.27.5.6 LoginToSafetyOfflineProgram

Program code

```
if (!safetyAdministration.IsLoggedOnToSafetyOfflineProgram)
{
    SecureString password = new NetworkCredential("",
"password").SecurePassword;
    safetyAdministration.LoginToSafetyOfflineProgram(password);
}
```

Application

The action `LoginToSafetyOfflineProgram` logs-in the user into the safety program if the passed `password` is correct. After a successful login changes to the safety program can be made.

The user gets feedback in the form of an exception in the following cases.

- A safety program password is not set
- The password is null
- The password is wrong
- If the safety program cannot be changed (see "SafetyModificationsPossible (Page 1191)"), or the project is UMAC protected, manipulating the safety program password via Openness is not allowed.

- The user is not offline
- Safety license is not present

5.27.5.7 LogoffFromSafetyOfflineProgram

Program code

```
if (safetyAdministration.IsLoggedOnToSafetyOfflineProgram)
{
    safetyAdministration.LogoffFromSafetyOfflineProgram();
}
```

Application

This action `LogoffFromSafetyOfflineProgram` the user from the safety program. Thus, it's not possible to make changes to the safety program anymore.

The user gets feedback in the form of an exception in the following cases.

- A safety program password is not set
- If the safety program cannot be changed (see "SafetyModificationsPossible (Page 1191)"), or the project is UMAC protected, manipulating the safety program password via Openness is not allowed.
- The user is not offline
- Safety license is not present

5.27.5.8 Runtime Groups

Create a new Runtime Group

Program code

```
// Creates a runtime group with the passed name and creates a main
safety FB and main safety IDB.
RuntimeGroup RuntimeGroups.Create("F-runtime group 2");

// Creates a runtime group with the passed FC for the main safety
block.
PlcBlock mainSafetyFC;
RuntimeGroup RuntimeGroups.Create("F-runtime group 2",
mainSafetyFC);

// Creates a runtime group with the passed FB and IDB for the main
safety block.
PlcBlock mainSafetyFB;
PlcBlock mainSafetyIDB;
```

```
RuntimeGroup RuntimeGroups.Create("F-runtime group 2",
mainSafetyFB, mainSafetyIDB);
```

Application

Allows to create a new runtime group with the specified parameters.

An exception is thrown:

- if the passed name is invalid.
- if a runtime group with the given name already exists.
- if the maximum number of allowed runtime groups is exceeded.
- if the passed main safety blocks are invalid or of the wrong type.

Delete a Runtime Group

Program code

```
runtimeGroup.Delete();
```

Application

This method call deletes the runtime group. The referenced blocks in the runtime group are not deleted.

Runtime Groups

Program code

```
foreach (RuntimeGroup runtimeGroup in
safetyAdministration.RuntimeGroups)
{
    // access the properties on the RuntimeGroup object...
}

RuntimeGroup runtimeGroup1 =
safetyAdministration.RuntimeGroups.Find("RTG1");
```

Application

The property `RuntimeGroups` returns all runtime groups. Furthermore, one specific runtime group can be retrieved by using the `Find` method. The `Find` method returns null if a runtime group with the given name does not exist.

The following properties can be accessed from a `RuntimeGroup` instance at hand.

Runtime group name

Program code

```
string name = runtimeGroup.Name;
```

Application

Returns the name of the runtime group.

F-OB Name

Program code

```
string name = runtimeGroup.FOBName;
```

Application

Returns the F-OB name or null if no F-OB is set.

F-OB EventClass

Program code

```
string eventClass = runtimeGroup.FOBEventClass;
```

Application

Returns the F-OB event class or null if no F-OB is set.

F-OB Number

Program code

```
Int32 number = (Int32) runtimeGroup.GetAttribute("FOBNumber");
runtimeGroup.SetAttribute("FOBNumber", 123);
```

Application

Returns and sets the F-OB number. If the F-OB number is not available, null is returned. If the value is set and no F-OB is available, or the value is not within the valid range, an exception is thrown.

F-OB Cycle Time

Program code

```
Int32 cycleTime = (Int32)
runtimeGroup.GetAttribute("FOBCycleTime");
runtimeGroup.SetAttribute("FOBCycleTime", 100000);
```

Application

Returns and sets the cycle time of the runtime group. If the value is not available, null is returned. If the set value is not available or within the valid range, an exception is thrown.

F-OB Phase shift

Program code

```
Int32 phaseShift = (Int32)
runtimeGroup.GetAttribute("FOBPhaseShift");
runtimeGroup.SetAttribute("FOBPhaseShift", 120);
```

Application

Returns and sets the Fail-safe organization block's phase shift of the runtime group. If the value is not available, null is returned. If the set value is not available or not within the valid range, an exception is thrown.

F-OB Priority

Program code

```
Int32 priority = (Int32) runtimeGroup.GetAttribute("FOBPriority");
runtimeGroup.SetAttribute("FOBPriority", 10);
```

Application

Returns and sets the Fail-safe organization block's priority of the runtime group. If the value is not available, null is returned. If the set value is not available or not within the valid range, an exception is thrown.

Main safety block name

Program code

```
string mainSafetyName = runtimeGroup.MainSafetyBlockName;
```

Application

Returns the name of the main safety block or null if no block is set.

Main safety block IDB name

Program code

```
string mainSafetyIDbName = runtimeGroup.MainSafetyBlockIDbName;
```

Application

Returns the name of the main safety's Instance DB or null if no block is set.

Warn Cycle Time

Program code

```
Int32 warnCycleTime = runtimeGroup.WarnCycleTime;
runtimeGroup.WarnCycleTime = 110000;
```

Application

Returns and sets the warn cycle time of the runtime group. If the set value is not within the valid range, an exception is thrown.

Maximum Cycle Time

Program code

```
Int32 maximumCycleTime = runtimeGroup.MaximumCycleTime;
runtimeGroup.MaximumCycleTime= 120000;
```

Application

Returns and sets the maximum cycle time of the runtime group. If the set value is not within the valid range, an exception is thrown.

F-runtime group information DB

Program code

```
string infoDbName = runtimeGroup.InfoDbName;
runtimeGroup.InfoDbName = "RTGSysInfo";
```

Application

Returns and sets the name of the already existing information DB (RTGSysInfoDB) of the runtime group. This method call does not create a new DB. If the information DB name is set and invalid, an exception is thrown.

Pre Processing

Program code

```
string preProcessingName = runtimeGroup.PreProcessingName;
runtimeGroup.PreProcessingName = "PreProcessing";
```

Application

Returns and sets the name of the pre processing FC. If the pre processing FC is not set, null is returned. If the pre processing name is set and invalid, an exception is thrown. Setting the name to empty or null resets the pre processing, but does not delete the referenced FC.

Post Processing

Program code

```
string postProcessingName = runtimeGroup.PostProcessingName;
runtimeGroup.PostProcessingName = "PostProcessing";
```

Application

Returns and sets the name of the post processing FC. If the post processing FC is not set, null is returned. If the post processing name is set and invalid, an exception is thrown. Setting the name to empty or null resets the post processing, but does not delete the referenced FC.

Generate global F-I/O status block

Program code

```
PlcBlock block = runtimegroup.GenerateGlobalFIOStatusBlock();
```

Application

Creates a global F-I/O status block or overrides it if the status block already exists.

See also: "SIMATIC Safety - Configuring and Programming (https://support.industry.siemens.com/cs/ww/en/view/54110126)" manual in chapter "Creating a global F-I/O status block"

5.27.5.9 Safety settings

Safety settings

Program code

```
SafetySettings settings = safetyAdministration.Settings;
```

Application

The property `SafetySettings` returns a container for the following properties that can be accessed from the returned instance.

Assignment of block numbers generated by the safety system

Program code

```
AssignmentOfBlockNumbers blockNumbers =
settings.AssignmentOfBlockNumbers;
ushort fromFBNumber = blockNumbers.FromFB;
ushort toFBNumber = blockNumbers.ToFB;
ushort fromFCNumber = blockNumbers.FromFC;
ushort toFCNumber = blockNumbers.ToFC;
ushort fromDBNumber = blockNumbers.FromDB;
ushort toDBNumber = blockNumbers.ToDB;

BlockNumbersManagementMode mode = blockNumbers.ManagementMode;
if (mode != BlockNumbersManagementMode.FSystemManaged) {
    // Set block numbers...
}
```

Application

Gets or sets the block numbers generated by the safety system. If a number is set and the value is not within the valid range, an exception is thrown. If a number is set and the management mode is `FSystemManaged`, an exception is thrown.

Safety System Version

Program code

```
SafetySystemVersion version = settings.SafetySystemVersion;
IList<SafetySystemVersion> versions =
settings.GetApplicableSafetySystemVersions();

settings.SafetySystemVersion = versions.Last();
```

```
SafetySystemVersion version1_6 = versions.SingleOrDefault(v =>
v.Value == "V1.6");
if (version1_6 != null) {
    settings.SafetySystemVersion = version1_6;
}
```

Application

Gets or sets the Safety system version of the F-CPU.

With `GetApplicableSafetySystemVersions()` a list of the applicable Safety system versions for the corresponding F-CPU can be fetched. The received list is sorted in ascending order by version number. With `.Last()` the highest available version is received.

A specific version can also be fetched by selecting an item from the applicable Safety system version list.

Safety mode can be disabled

Program code

```
bool safetyModeCanBeDisabled = settings.SafetyModeCanBeDisabled;
settings.SafetyModeCanBeDisabled = true;
```

Application

Gets or sets whether the Safety mode can be disabled.

Activation of F-change history

Program code

```
bool activationOfFChangeHistory =
settings.ActivationOfFChangeHistory;
settings.ActivationOfFChangeHistory = true;
```

Application

Gets or sets the activation of the F-change history.

Enable of consistent upload

Program code

```
bool consistentUpload = (bool)
settings.GetAttribute("EnableConsistentUploadFromFCpu");
settings.SetAttribute("EnableConsistentUploadFromFCpu", true);
```

Application

Gets or sets whether the consistent upload from a F-CPU is enabled. This setting is only available for read and write on specific F-CPUs.

Enable F-Communication ID tag

Program code

```
bool fCommunicationIdTag = (bool)
settings.GetAttribute("EnableFCommunicationIdTag");
settings.SetAttribute("EnableFCommunicationIdTag", true);
```

Application

Gets or sets whether the F-Communication ID tag is enabled.

For safety assessment of this information refer to the manual "SIMATIC Safety – Configuring and Programming".

Create driver instance data blocks without prefix

Program code

```
bool prefix = settings.CreateDriverInstanceDataBlocksWithoutPrefix;
settings.CreateDriverInstanceDataBlocksWithoutPrefix = true;
```

Application

Gets or sets whether the names of the F-I/O instance DBs are created without prefix.

Clean up of system generated objects

Program code

```
settings.CleanSystemGeneratedObjects();
```

Application

Cleans up the system generated objects by the compile.

5.27.6 SafetyPrintout

This feature is available for S7-1200/1500 F-CPUs.

Program code

Obtain the `SafetyPrintout` service from a `DeviceItem` instance at hand. Then call the `Print()` function on this service. This function returns true on success.

```
SafetyPrintout printoutService =
deviceItem.GetService<SafetyPrintout>();
if (printoutService != null) {
    FileInfo path = new FileInfo(@"C:\safety_printout.pdf");
    bool success =
printoutService.Print(SafetyPrintoutFilePrinter.MicrosoftPrintToPdf,
 path, "DocuInfo_ISO_A4_Portrait", SafetyPrintoutOption.All)
}
```

Application

The action `Print(...)` of the service `SafetyPrintout` is used to trigger the safety printout with the given parameters. The given function returns true on a successful printout.

The given path needs to be valid in order to save the printout.

For the printout a given `documentLayout` (e.g. "DocuInfo_ISO_A4_Portrait") can be set. Furthermore, a All or Compact safety printout is selectable with the `documentationOption` parameter.

Furthermore, the selected digital printer (`SafetyPrintoutFilePrinter` enum) in the system is used to create the digital safety printout. The "Microsoft Print to PDF" or "Microsoft XPS Document Writer" printer can be selected. These printers are available and are activated by default on Windows 10 ("Microsoft XPS Document Writer" since Windows 7). For the printout to succeed, the selected printer needs to be activated. If this is not the case, these printers can be enabled in the Windows feature window by executing the following file after pressing the *Windows + R* keys.

```
optionalfeatures.exe
```

Export/import

6

6.1 Overview

6.1.1 Basic principles of importing/exporting

Introduction

You can export certain configuration data and then re-import the data to the same or a different project after editing.

Note

There are no obligations or guarantees of any kind associated with using this description to manually modify and evaluate the source file. Siemens therefore accepts no liability arising from the use of all or part of this description.

Exportable and importable objects

The following configuration data can also be imported or exported by means of TIA Portal Openness APIs:

Table 6-1 Projects

Objects	Export	Import
Project graphics	X	X

Table 6-2 PLC

Objects	Export	Import
Blocks	X	X
Know-how protected blocks	X	–
Failsafe blocks	X	X
System blocks	X	–
PLC tag tables	X	X
Technology objects	X	X
PLC tags and constants	X	X
User data types	X	X
DBs	X	-

Table 6-3 HMI

Objects	Export	Import
Screens	X	X
Screen templates	X	X
Global screens	X	X
Pop-up sreens	X	X
Slide-in screens	X	X
Scripts	X	X
Text lists	X	X
Graphic lists	X	X
Cycles	X	X
Connections	X	X
Tag table	X	X
Tags	X	X

Complete export or export of open references

The object types listed above are exported or imported along with all objects if these belong to the same sub-tree. This rule is also valid for referenced objects of the same sub-tree.

For referenced objects in other sub-trees, however, a complete export or import is not possible. Instead, "open references" to these objects are exported or imported.

Referenced objects of the same sub-tree are only exported if they belong to the group of exportable objects. Any dynamizations on objects are treated as objects during the import/ export, and are exported and imported as well.

The export includes all object attributes that were changed during configuration. This applies regardless of whether the altered attribute will be used or not.

Example: You have configured a graphic IO field with the mode "Input/Output" and selected the setting "Visible after clicking" for the attribute "Scroll bar type". In the course of configuration, you have changed the mode to "Two states". The attribute "Scroll bar type" is not available in this mode. Because the "Scroll bar type" attribute was changed, it is included in the export, even though the attribute is not used.

Importing open references

You can also import objects with open references (see Importing configuration data (Page 1212)).

If the referenced objects are contained in the target project, the open references are automatically linked to the object types again. These objects must be available at the same location and be assigned to the same name as for the export. If the referenced objects are not contained in the target project the open references can't be resolved. No additional object will be created to resolve these open references.

Export and import file format

The export and import file format is XML. Only CAx data require AML format. The different schema definitions for all formats are described in the respective section of this manual:

- XML format for the data of a HMI devices (Page 1221)
- XML format for the data of a PLC devices (Page 1295)
- AML format for CAx data (Page 1431)

Importing and exporting fonts

Fonts defined on objects are also exported and imported.

When you import fonts that are not included in the project, the standard font is displayed at the object after the import. However, the imported font is stored in the data management.

If the attributes for a font are not assigned in an import file, the attributes are assigned default values after the import.

Importing and exporting technology objects

You can find out which technology objects can be exported and imported as of TIA Portal version V16 you can find in chapter "Overview of technology objects and versions".

Restrictions

The export format is internal and valid exclusively for the current version of TIA Portal Openness. The export format may change in future versions.

All errors which occur during the import and export are reported as exceptions.
For more information on exceptions, see section Handling exceptions (Page 1187).

See also

Field of application for Import/Export (Page 1209)

Exporting configuration data (Page 1211)

6.1.2 Field of application for Import/Export

Introduction

The Import/Export functionality allows you to export specific objects in a targeted manner.

You can edit the exported data in an external program, or reuse it unchanged in other TIA Portal projects.

If you structure the import file correctly, you can also import configuration data created externally without having to carry out an export first.

Note

If you import externally created configuration data which contain code errors or a wrong structure this could cause unexpected errors.

Field of application

Exporting and importing data is useful for the following tasks:

- For externally editing configuration data.
- For importing externally-created configuration data, e.g. text lists and tags.
- For distributing specified configuration data to different projects, e.g. a modified process screen which is to be used in several projects.
- For replicating and adjusting the hardware configuration between the TIA Portal project and an ECAD program.

See also

Basic principles of importing/exporting (Page 1207)

6.1.3 Version Specific Simatic ML Import

Introduction

Each version of the Openness API supports export of Simatic ML files. The version of the exported Simatic ML file should match with the version of the TIA Portal rather than that of the Openness API version .

The SimaticML file version for export always corresponds with the used TIA Portal version, regardless which Openness API version is used. Until TIA Portal V18, importing SimaticML files is only possible, if the used Openness API version is higher or equal to the SimaticML version.
Since TIA Portal V19, all SimaticML versions can be imported with all Openness API versions as long as the SimaticML version is within the LTS scope (means from Vxx-3 to Vxx). This enables long-term stable roundtrips.

Note

Vxx refers to the current installed version of TIA Portal.

6.1.4 Editing the XML file

Introduction

Use an XML editor or text editor for editing an XML file for importing configuration data.

If you are making comprehensive changes or are creating custom object structures, we recommend that you use an XML editor with auto-complete function.

Note

Changing the XML content requires comprehensive knowledge of the structure and validation rules in XML. Work manually in the XML structure only in exceptional cases in order to avoid validation errors.

6.1.5 Exporting configuration data

Introduction

The configuration data of each start object (root) is exported separately to an XML file.

Editing the export file requires an adequate knowledge of XML. Use an XML editor for more convenient editing.

Example

You have a process screen that contains an IO field. An external tag is configured in the IO field. An export of the process screen includes the screen and the IO field. The tag and the connection used by the tag are not exported. Instead, only an open reference is included in the export.

Contents of the export file

Beginning with the start object, all objects of a sub-tree and their attributes are saved to the export file. All references to objects of different sub-trees are only exported as open references. The corresponding attributes of the referenced objects in different sub-trees are not written to the export file.

Note

Export of object types from the library is not supported

You can create objects as a type in the library. Instances of the object type used in the project can be edited like other objects using the TIA Portal Openness application. When you export objects, the instances are exported without the type information.

When you re-import these objects into the project, the instances of the object types are overwritten and the instance is detached from the object type.

The export file does not necessarily contain all the attributes of an object. You define what data is to be exported:

- `ExportOptions.None`
  This setting exports only the modified data or the data that differs from the default.
  The export file also contains all values that are obligatory for the subsequent data import.
- `ExportOptions.WithDefaults`[1]
  The default values are also exported.
- `ExportOptions.WithReadOnly`[1]
  The write-protected values are also exported.

[1]: You can combine these two options with the following syntax: `Export(path,ExportOptions.WithDefaults | ExportOptions.WithReadOnly);`

The entire contents of the export file are in English. Regardless of this, any project texts contained are exported and imported in all the languages present.

All configuration data is modeled as XML objects in the export file.

See also

Basic principles of importing/exporting (Page 1207)

Exporting blocks (Page 1351)

6.1.6 Importing configuration data

Introduction

Configuration data is imported from a previously exported and edited XML file or from an XML file you have created yourself. The data contained in this file is checked during the import. This approach prevents the configuration data in the TIA Portal from becoming inconsistent as a result of the import.

Restrictions

- All root objects in the import file have to be of the same kind, e.g. tag tables, blocks.
- If an import file includes several root objects and one of them is not valid, the entire contents of the import file are not imported.
- When importing texts, the corresponding project languages must have been set up in the target project in order to exclude import failure. If necessary you can modify the language settings via TIA Portal Openness or use new culture specific enum options during Import to avoid import failure.
- The culture specific enum options ImportOptions.ActivateInactiveCultures and Importoptions.SkipInactiveCultures are not supported for any excel based import APIs (For example: 'ImportSupervisionSettingsFromXlsx'). Usage of these options will result in import failure.

- The import will fail when importing texts, if the corresponding project languages are not supported in TIA Portal.
- If you specify invalid attributes of an object in the import file that cannot be edited in the graphical user interface of TIA Portal, the import is canceled.
- Only the area pointers listed in the "separately for each connection" field can be imported or exported.
- The import of object types from the library is not supported. You can create objects as a type in the library. Instances of the object type used in the project can be edited like other objects using the TIA Portal Openness application. When you export objects, the instances are exported without the type information. When you re-import these objects into the project, the instances of the object types are overwritten and the instance is detached from the object type.

Note

Device-dependent value ranges for graphical attributes

If values of graphical attributes exceed the valid value range, these values are reset to the possible maximum values for the HMI device during import.

Different import behavior

If objects to be imported already exist in the project, control the import behavior using different program codes. Otherwise, the objects will be created again in the project during the import.

The following settings for the import behavior are possible:

- `ImportOptions.None`
  By using this setting configuration data will be imported without overwritting.
  If an object being imported from an XML file which already exists in the project, the import is interrupted and an exception will be thrown.
- `ImportOptions.Override`
  By using this setting configuration data will be imported with automatic overwritting.
  You can specify that existing objects in the project are overwritten with the import. Relevant objects are deleted prior to the import and recreated with default values. These defaults are overwritten with the imported values during the import. If the existing object and the new object are not in the the same group overwriting can't take place. To avoid naming conflicts import is canceled and an exception is thrown.
- `ImportOptions.SkipInactiveCultures`
  By using this setting configuration data will be imported by skipping the text for which required project languages are not active in the project.
- `ImportOptions.ActivateInactiveCultures`
  By using this setting configuration data will be imported with an automatic activation of required project languages corresponding to importing text if not activated already in the project.

Note

If you want combined import behavior, options can be used in combination (using Bitwise OR operator). However, you cannot combine ActivateInactiveCultures and SkipInactiveCultures options which are contradicting to each other. This combination will result in import failure. Usage of Bitwise AND in combination will result in ImportOptions.None irrespective of any combination used.

Procedure for importing

If you wish to import an XML file, the data it contains must adhere to certain rules. The contents of the import file must be well-formed. There must be no syntax errors and no data structure errors. In the case of comprehensive changes, use an XML editor that checks these criteria prior to the import.

During the import of the XML file to the TIA Portal, the data it contains is first checked for formal errors in the XML code. If errors are detected during the check, the import is canceled and the errors are shown in an exception (see Handling exceptions (Page 1187)).

See also

Basic principles of importing/exporting (Page 1207)

Importing user data type (Page 1387)

6.2 Import/export of project data

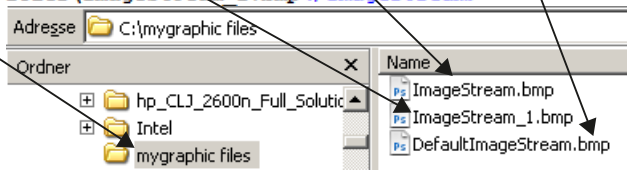
6.2.1 Project graphics

6.2.1.1 Exporting/importing graphics

Introduction

The export of configuration data from the TIA Portal to the XML file does not include selected graphics, or graphics referenced by an object. The graphics are saved separately during the export. In the XML file, the graphics are referenced by a relative path and their file name. A graphic reference is modeled in the XML file as an object and contains an attribute list and, if necessary, a link list, just like the other objects.

```
<Hmi.Globalization.MultiLingualGraphic ID="0">
    <AttributeList>
      <DefaultDithering>False</DefaultDithering>
      <DefaultImageStream external="path">mygraphic files\DefaultImageStream.bmp</DefaultImageStream>
      <DefaultSmoothness>False</DefaultSmoothness>
      <Name>MyGraphic1</Name>
    </AttributeList>
    <ObjectList>
      <Hmi.Globalization.GraphicItem ID="1" CompositionName="Items">
        <AttributeList>
          <Culture>en-US</Culture>
          <Dithering>False</Dithering>
          <ImageStream external="path">mygraphic files\ImageStream.bmp</ImageStream>
          <Smoothness>False</Smoothness>
        </AttributeList>
      </Hmi.Globalization.GraphicItem>
      <Hmi.Globalization.GraphicItem ID="2" CompositionName="Items">
        <AttributeList>
          <Culture>de-DE</Culture>
          <Dithering>False</Dithering>
          <ImageStream external="path">mygraphic files\ImageStream_1.bmp</ImageStream>
          <Smoothness>False</Smoothness>
        </AttributeList>
      </Hmi.Globalization.GraphicItem>
    </ObjectList>
  </Hmi.Globalization.MultiLingualGraphic>
```



Exporting graphics

The export of configuration data includes only graphics that were selected directly for export. The exportable graphics are stored in the TIA Portal for the specific language. If a project is configured in multiple languages, all the language versions used are exported.

When you export graphics, a new folder is created in the export file folder. The file folder name is built by concatenating the xml-filename with " files". This folder contains the

exported graphics. If this folder exists already, a new folder is created and supplemented by a consecutive number.

The graphics are saved in the same file format as in the project. The data format is not changed or converted, and the resolution and color depth remain unchanged.

The ID "default" is used as the file extension for the language selected as the default language.

If the folder already contains a file of the same name, the file name of the exported graphic is supplemented by a consecutive number.

Importing graphics

The following requirements apply when importing graphics:

- The graphics must have a file format that is supported by TIA Portal.
- The graphics must be referenced in the XML file by a relative path specification.

Once you have exported a graphic, you can edit it outside TIA Portal using a graphics program and then re-import it.

See also

Basic principles of importing/exporting (Page 1207)

6.2.1.2 Exporting all graphics of a project

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

You can export either a single graphic or all graphics of the graphics collection of a project in all languages. An XML file with all project graphic entries concerned is created during the export and referenced along with the exported graphics. The relevant graphics are saved along with the XML file to the same directory of the file system.

To allow the exported graphics ("*.jpg", "*.bmp", "*.png", "*.ico", etc.) to be changed, these graphics are not write-protected.

Program code: Exporting a graphic

Modify the following program code to export the required graphic:

```
//Exports all language variants of a single grafic
Project project = …;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
MultiLingualGraphic graphic = graphicsComposition.Find("graphicName");
graphic.Export(new FileInfo(@"D:\ExportFolder\graphicName.xml"),
ExportOptions.WithDefaults);
```

Program code: Exporting all graphics

Modify the following program code to export all graphics of a graphics collection:

```
//Exports all graphics of a graphic library
Project project = …;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
foreach(MultiLingualGraphic graphic in graphicsComposition)
{
    graphic.Export(new FileInfo(string.Format(@"D:\Graphics\{0}.xml", graphic.Name)),
ExportOptions.WithDefaults);
}
```

6.2.1.3 Importing graphics to a project

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

An XML file is saved along with the language versions of a graphic to a directory of your file system.

You can reference all graphics in a relative path in the XML file.

You can now import all language versions of a graphic contained in the XML file to the graphics collection.

You should also observe the Importing configuration data (Page 1212).

Program code

Modify the following program code to import one or several graphics:

```
//Import all language variants of a single graphic
Project project = …;
MultiLingualGraphicComposition graphicComposition = project.Graphics;
graphicComposition.Import(new FileInfo(@"D:\Graphics\Graphic1.xml"),
ImportOptions.Override);
```

6.2.2 Project texts

6.2.2.1 Export of project texts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

In TIA Portal you can find project texts below the "Languages & resources" node of a project. These texts are exported to a "*.xlsx" file which is used for example for translations. The limitations of exporting and importing project texts are the same as in the UI. These limitations include:

- Exported texts can only be imported into the project from where they were exported.
- You can only translate texts to languages that are available in in the project. If necessary you can add project languages via TIA Portal Openness.
- Only existing texts can be re-imported, if text from the original project has been deleted or re-created the import for that text will fail.

You have to define the following parameters:

Name	Example	Description
pah	new FileInfo ("D:\Test\Project-Text.xlsx")	Path to export file
sourceLanguage	new CultureInfo("en-US")	Reference language text is to be translated from
targetLanguage	new CultureInfo("de-DE")	Target language text is to be translated to

Note

Multilingual texts will be exported together with the parent object to which they belong. Multilingual texts can not be exported explicitly.

Program code: Export from "Languages & resources" node

The use of the example parameters leads to the following program code to export project texts:

```
project.ExportProjectTexts(new FileInfo(@"D:\Test\ProjectText.xlsx"), new CultureInfo("en-US"), new CultureInfo("de-DE"));
```

XML structure of a exported multilingual text item

```
...
  <MultilingualText ID="2" CompositionName="Comment">
    <ObjectList>
      <MultilingualTextItem ID="3" CompositionName="Items">
        <AttributeList>
          <Culture>en-US</Culture>
          <Text>My super tag</Text>
        </AttributeList>
      </MultilingualTextItem>
      <MultilingualTextItem ID="4" CompositionName="Items">
        <AttributeList>
          <Culture>ru-RU</Culture>
          <Text>Мой супер тэг</Text>
        </AttributeList>
      </MultilingualTextItem>
    </ObjectList>
  </MultilingualText>
...
```

6.2.2.2 Import of project texts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

In TIA Portal you can find project texts below the "Languages & resources" node of a project. You can import project texts from a ".xlsx" file which is used for example for translations. The limitations of exporting and importing project texts are the same as in the UI. These limitations include:

- Exported texts can only be imported into the project from where they were exported.
- After a project was saved under a new name using "Save as ...", the import of texts is not possible anymore.
- You can only import translated texts in languages that are available in project from where they were exported.
- Only existing texts can be re-imported, if text from the original project has been deleted or re-created the import for that text will fail.

You have to define the following parameters:

Name	Example	Description
path	new FileInfo(@"D:\Test\Project-Text.xlsx")	Path to import file
updateSourceLan-guage	true	If true, the text of the reference language is updated from the export file. If false, the text of the reference language is not updated

Note

Multilingual texts will be imported together with the parent object to which they belong. Multilingual texts can not be imported explicitly.

Program code

The use of the example parameters leads to the following program code to import project texts:

```
ProjectTextResult result = project.ImportProjectTexts(new
FileInfo(@"D:\Test\ProjectText.xlsx"), true);
```

The import of the Project Texts returns an object indicating the status of the Import and path to which the import log is saved. These attributes can be accessed with the following code:

```
ProjectTextResultState resultState = result.State;
FileInfo logFilePath = result.Path;
```

6.3 Importing/exporting data of an HMI device

6.3.1 Data structure for import/export

6.3.1.1 Structure of an XML file

Introduction

The data in the export file from the import/export is structured with reference to a basic structure.

Basic structure of an export file

The export file is generated in a XML format.

The XML file starts with a document information. It includes the data of the computer-specific installation with which the project was exported.

The export file is divided into the following two sections:

- Information about the document
  In this section, you can enter your own information about the export in valid XML syntax. The content is ignored by the import.
  For example you can insert a `<IntegrityInformation>...</IntegrityInformation>` block, in which you place additional information about the validation. After the XML file is forwarded, the recipient can use this block before the import to check whether the XML file has been changed.
- Object
  This section contains the elements to be exported.

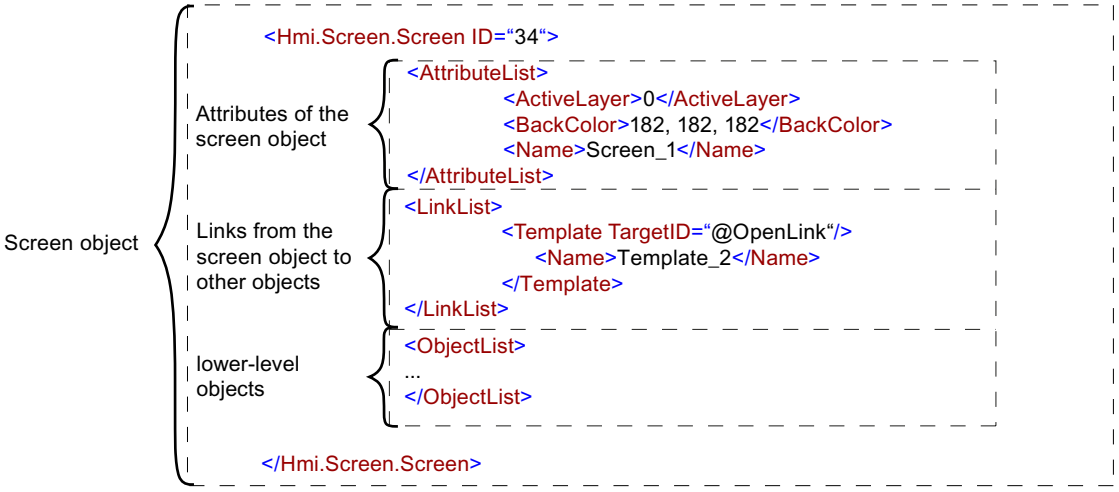
Information about the document

Screen object

```
<?xml version="1.0" encoding="UTF-8" ?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <DocumentInfo>
        <UserName>Jane Doe</UserName>
        <Company>Example_Inc</Company>
        <IntegrityInformation>...</IntegrityInformation>
        <Created>2016-04-28T18:05:42.179207Z</Created>
        <ExportSetting>WithDefaults</ExportSetting>
        <InstalledProducts>
            <Product>
                <DisplayName>Totally Integrated Automation Portal</DisplayName>
                <DisplayVersion>V14</DisplayVersion>
            </Product>
            <OptionPackage>
                <DisplayName>WinCC Professional</DisplayName>
                <DisplayVersion>V14</DisplayVersion>
            </OptionPackage>
            <OptionPackage>
                <DisplayName>Siemens TIA Openness</DisplayName>
                <DisplayVersion>V14</DisplayVersion>
            </OptionPackage>
        </InstalledProducts>
    </DocumentInfo>
    <Hmi.Screen.Screen ID ="0">
        <AttributeList>
            <ActiveLayer>0</ActiveLayer>
            <BackColor>189,190,0</BackColor>
            <Height>422</Height>
            <Name>Root screen</Name>
            <Number>1</Number>
            <Visible>True</Visible>
            <Width>640</Width>
        </AttributeList>
        <LinkList>
            <Template TargetID="@OpenLink">
                <Name>Template_1</Name>
            </Template>
        </LinkList>
        <ObjectList>
            <Name>dummy</Name>
        </ObjectList>
    </Hmi.Screen.Screen>
</Document>
```

Screen objects of an export file

The exported elements are available in additional elements of the XML file.



See also

Basic principles of importing/exporting (Page 1207)

6.3.1.2 Structure of the data for importing/exporting

Objects

The basic structure is the same for all objects.

Every object in the XML file starts with its type, for example, "Hmi.Screen.Button", and an ID. The ID is created automatically during export.

```
<Hmi.Screen.Button CompositionName="ScreenItems" ID="60">
```

Each object apart from the start object also contains an "CompositionName" XML attribute. The value for this attribute is preset. It is occasionally necessary to specify this attribute, for example, to change the label when a button is pressed or released.

```
<MultilingualText ID="A" CompositionName="TextOff">
    <ObjectList>
        <MultilingualTextItem ID="B" CompositionName="Items">
            <AttributeList>
                <Culture>en-US</Culture>
                <Text>
                    <body>
                        <p>TextOff</p>
                    </body>
                </Text>
            </AttributeList>
        </MultilingualTextItem>
    </ObjectList>
</MultilingualText>
<MultilingualText ID="C" CompositionName="TextOn">
    <ObjectList>
        <MultilingualTextItem ID="D" CompositionName="Items">
            <AttributeList>
                <Culture>en-US</Culture>
                <Text>
                    <body>
                        <p>TextOn</p>
                    </body>
                </Text>
            </AttributeList>
        </MultilingualTextItem>
    </ObjectList>
</MultilingualText>
```

Attributes

Every object contains attributes that are contained in an "AttributeList" section. Every attribute is modeled as an XML element, e.g. "BackColor". The value of an attribute is modeled as XML content, e.g. "204, 204, 204".

```
<Hmi.Screen.Button ID="2" CompositionName="ScreenItems">
    <AttributeList>
        <BackColor>204, 204, 204</BackColor>
        <ObjectName>Button_1</ObjectName>
    </AttributeList>
</Hmi.Screen.Button>
```

For referencing objects, each object contains a "LinkList" section, if necessary. This section contains links to other objects inside or outside the XML file. Every link is modeled as an XML element. The designation of a link is defined by the target object in the schema file. Every link also contains the "TargetID" attribute. When the target object is included in the XML file, the value of the "TargetID" attribute is the ID of the referenced object preceded by a "#". When the target object is not included in the XML file, the value of the "TargetID" attribute is "@OpenLink". The actual reference to the object is modeled as subordinate XML element.

```
<Hmi.Tag.Tag ID="17">
    <AttributeList>
        <Name>Tag_1</Name>
    </AttributeList>
    <LinkList>
        <AcquisitionCycle TargetID="@OpenLink">
            <Name>2 s</Name>
        </AcquisitionCycle>
        <Connection TargetID="@OpenLink">
            <Name>HMI_connection</Name>
        </Connection>
    </LinkList>
</Hmi.Tag.Tag>
```

Relation between objects and XML structure

The figures below show the relation between the exported XML structure and the associated objects in WinCC.

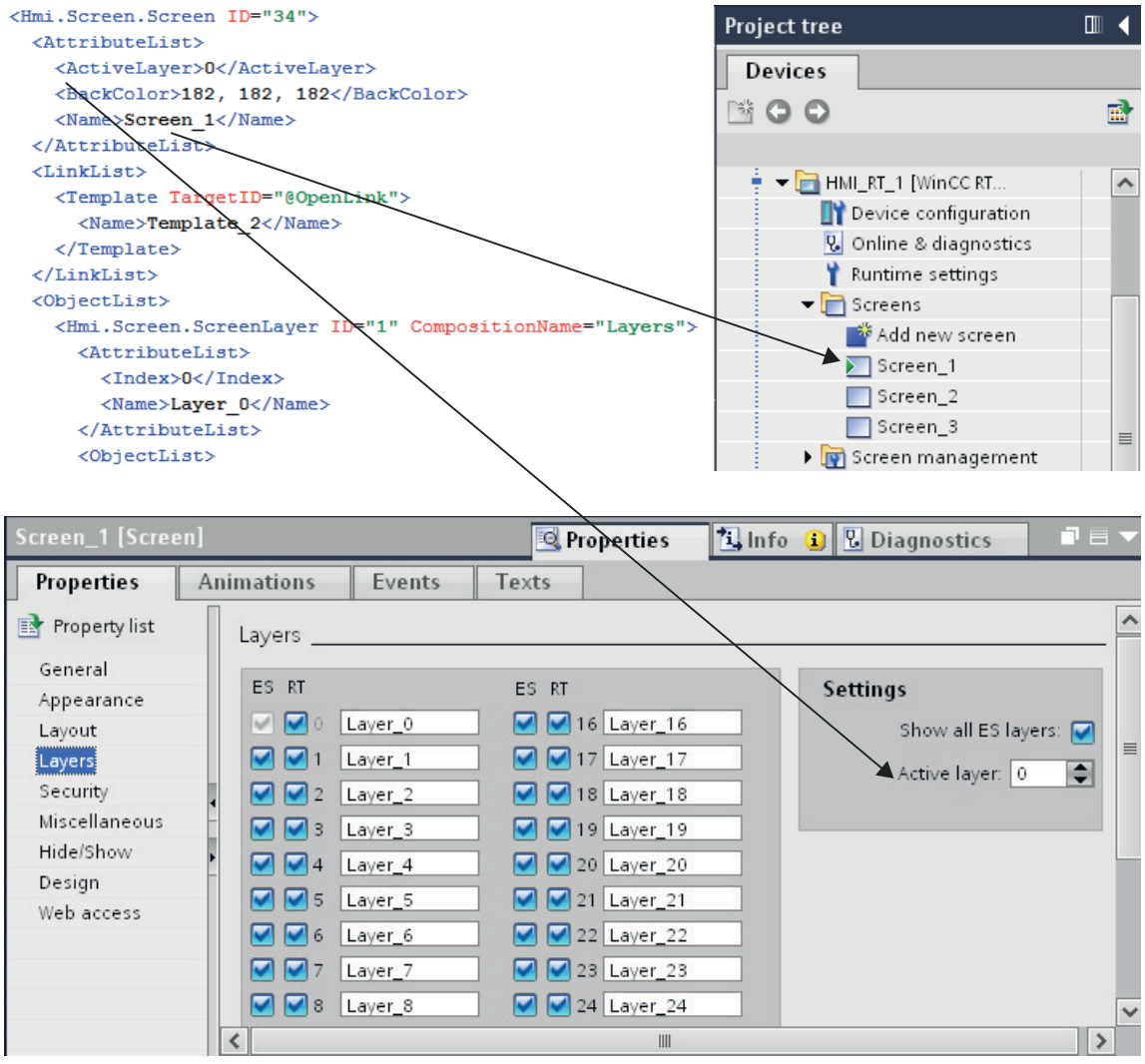


Figure 6-1 Relation between the WinCC user interface and the XML structure.

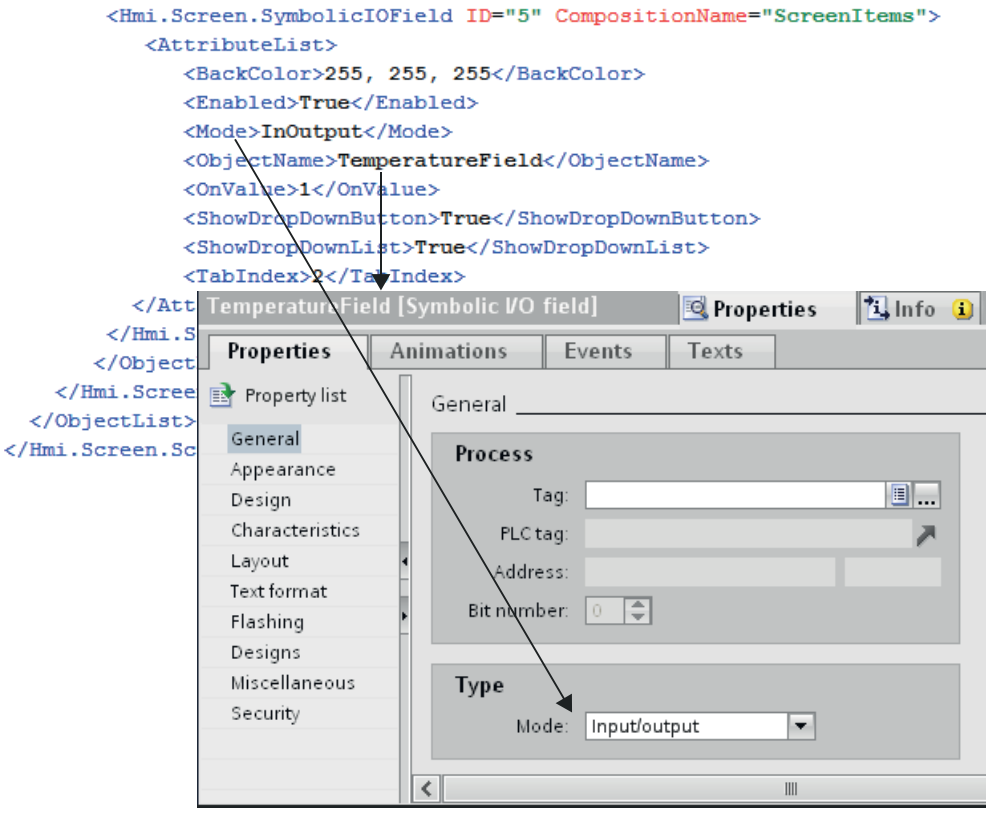


Figure 6-2 Relation between the settings in WinCC and the XML structure.

6.3.1.3 Cycles

Exporting cycles

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The TIA Portal Openness API interface supports the export of all cycles of a known HMI device to an XML file. The generation of the corresponding export file indicates that the export is complete.

Program code

Modify the following program code to export cycles from an HMI device to an XML file:

```
//Exports cycles from an HMI device
private static void ExportCyclesFromHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    foreach(Cycle cycle in cycles)
       {
           cycle.Export(new FileInfo(string.Format(@"C:\Samples\{0}.xml", cycle.Name)),
ExportOptions.WithDefaults);
       }
}
```

Importing cycles

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

When you use `ImportOptions.None`, you can identify the cycles that have actually been imported based on the composition number (Composition count). You have access to these imported cycles.

Note

Standard cycles have attributes that cannot be edited in the user interface. If you specify in the import file that these attributes should be changed, the import causes a NonRecoverableException and closes the TIA Portal.

Program code

Modify the following program code to import one or several cycles from an XML file into an HMI device:

```
//Imports cycles to an HMI device
private static void ImportCyclesToHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    string dirPathImport = @"C:\OpennessSamples\Import\";
    string cycleImportFileName = "CycleImport.xml";
    string fullFilePath = Path.Combine(dirPathImport, cycleImportFileName);

    cycles.Import(new FileInfo(fullFilePath), ImportOptions.None);
}
```

See also

Importing configuration data (Page 1212)

6.3.2 Tag tables

6.3.2.1 Exporting HMI tag tables

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

One XML file is exported per HMI tag table. The API supports this export process. The export of tag tables is also available in subfolders.

Program code: Exporting all HMI tag tables from a specified folder

Modify the following program code to export all HMI tag tables from a specific folder:

```
//Exports all tag tables from a tag folder
private static void ExportAllTagTablesFromTagFolder(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    foreach (TagTable table in tables)
    {
       FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name));
        table.Export(info, ExportOptions.WithDefaults);
    }
}
```

Program code: Exporting an HMI tag table

Modify the following program code to export an individual HMI tag table:

```
//Exports a tag table from an HMI device
private static void ExportTagTableFromHMITarget(HmiTarget hmitarget)
{
    string tableName = "Tag table XYZ";
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;
    TagTable table = tables.Find(tableName);

    if (table != null)
    {
       FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name));
        table.Export(info, ExportOptions.WithDefaults);
    }
}
```

Program code: Exporting all HMI tag tables

Modify the following program code to export all HMI tag tables:

```
//Exports all tag tables from an HMI device
private static void ExportAllTagTablesFromHMITarget(HmiTarget hmitarget)
{
    TagSystemFolder sysFolder = hmitarget.TagFolder;

    //First export the tables in underlying user folder
    foreach (TagUserFolder userFolder in sysFolder.Folders)
    {
        ExportUserFolderDeep(userFolder);
    }

    //then, export all tables in the system folder
    ExportTablesInSystemFolder(sysFolder);
}

private static void ExportUserFolderDeep(TagUserFolder rootUserFolder)
{
        foreach (TagUserFolder userFolder in rootUserFolder.Folders)
        {
            ExportUserFolderDeep(userFolder);
        }
        ExportTablesInUserFolder(rootUserFolder);
}

private static void ExportTablesInUserFolder(TagUserFolder folderToExport)
{
     TagTableComposition tables = folderToExport.TagTables;
     foreach (TagTable table in tables)
     {
         string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
         table.Export(new FileInfo(fullFilePath), ExportOptions.WithDefaults);
     }
 }

private static void ExportTablesInSystemFolder(TagSystemFolder folderToExport)
{
     TagTableComposition tables = folderToExport.TagTables;
     foreach (TagTable table in tables)
     {
         string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
         table.Export(new FileInfo(fullFilePath), ExportOptions.WithDefaults);
     }
 }
```

6.3.2.2 Importing HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to import the HMI tag table of an XML file to a user-defined folder or to a system folder:

```
//Imports a single HMI tag table from a XML file
private static void ImportSingleHMITagTable(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    FileInfo info = new FileInfo(@"D:\Samples\Import\myExportedTagTable.xml");
    tables.Import(info, ImportOptions.Override);
}
```

Incorrect import of tags

If you use the following symbols in the names of tags or referenced tags, the import of the tags will be faulty:

- . (period)
- \ (backslash)

Remedy 1:

Before an export, check that the name of the tag or referenced tag to be exported does not contain a period or backslash.

Remedy 2:

In the import file, exclude the names of tags or referenced tags using quotation marks.

Example

- Tag name with symbol:

  <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date\DB_SFX0908_HMI1.Actual_Date_Time.Hour</name>
- Tag name with symbol excluded in quotation marks:
  <name>"Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date\DB_SFX0908_HMI1.Actual_Date_Time.Hour"</name>

6.3.2.3 Exporting an individual tag from an HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following object model object types may possibly exist as sublevel items of an HMI tag and are taken into account during export:

MultilingualText	For Comment, TagValue, DisplayName
TagArrayMemberTag	For HMI array elements
TagStructureMember	For HMI structure elements
Event	For configured events
MultiplexEntry	For configured tag multiplexing entries

Program code

Modify the following program code to export an individual tag from an HMI tag table to an XML file:

```
//Exports a selected tag from a tag table
private static void ExportSelectedTagFromTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable mytable = tagFolder.TagTables.Find("MyTagTable");

    TagComposition containingTags = mytable.Tags;
    Tag myTag = containingTags.Find("MyTag");

    if (myTag != null)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Tags\{0}.xml",
myTag.Name));
        myTag.Export(info, ExportOptions.WithDefaults);
    }
}
```

6.3.2.4 Importing an individual tag into an HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following object model object types may possibly exist as sublevel items of an HMI tag and are taken into account during import:

MultilingualText	For Comment, TagValue, DisplayName
TagArrayMemberTag	For HMI array elements
TagStructureMember	For HMI structure elements
Event	For configured events
MultiplexEntry	For configured tag multiplexing entries

Program code

Modify the following program code to import an HMI tag from an XML file into an HMI tag table:

```
//Imports a tag into a tag table
private static void ImportTagIntoTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable myTable = tagFolder.DefaultTagTable;
    TagComposition tagComposition = myTable.Tags;

    FileInfo info = new FileInfo(@"D:\Samples\Import\myExportedTag.xml");
    tagComposition.Import(info, ImportOptions.Override);
}
```

6.3.2.5 Special considerations for the export/import of HMI tags

Introduction

Special considerations apply to the export and import of the following HMI tags:

- External HMI tags with integrated connection
- HMI tags with the "UDT" data type

Similar program codes

The program code for the above-mentioned HMI tags is almost identical to the following program codes:

- Program code: Exporting HMI tags (Page 1233)
- Program code: Importing HMI tags (Page 1234)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Special considerations for the export/import of an external HMI tag with integrated connection

When exporting an external HMI tag with integrated HMI connection, only the link of the HMI tag to the PLC tag is saved in the export file instead of the PLC tag data.

Before the import, you must ensure that the PLC, the corresponding PLC tags and the integrated connection to the corresponding PLC exist in the project. If this is not the case, these items must be created before the import. During the subsequent import of the external HMI tag, the link to the PLC tag will be activated again.

Names of external HMI tags must be unique across all tag tables of a project. If you do not specify the suitable tag table for the HMI tag during import, the import is canceled.

Use the following XML structure to import an external HMI tag with integrated connection:

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
    <AttributeList>
        <Name>MyIntegratedHmiTag_1</Name>
    </AttributeList>
    <LinkList>
        <AcquisitionCycle TargetID="@OpenLink">
            <Name>1 s</Name>
        </AcquisitionCycle>
        <Connection TargetID="@OpenLink">
            <Name>HMI_Connection_MP277_300400</Name>    <- Must exist in the project
        </Connection>
        <ControllerTag TargetID="@OpenLink">
            <Name>Datablock_1.DBElement1</Name>        <- Must exist in the project
        </ControllerTag>
    </LinkList>
</Hmi.Tag.Tag>
```

Note

The assignment of Structure Tags in the Upper & Lower Limit properties of the 'Elements of structure tags' is not supported in Comfort / Advanced devices. These structure tags can be assigned to the limit properties of Simple tags but is not supported to the limit properties of Structure tag Elements.

Special considerations for the export/import of an HMI tag of the "UDT" data type

The link is exported to the data type when an HMI tag of the "UDT" data type is exported. Only versioned data types are supported for import.

The data types must be saved in the project library. Data types in the global library are not supported.

The following rules apply to the import:

- The referenced data type must be contained in the project library.
  The import is terminated if the data type is not contained in the project library.
- The referenced data type must be versioned. Versioning is supported as of TIA Portal V13 SP1.
  An exception is thrown if the data type is not versioned.

  Note

  The first data type found is used during the import to resolve the reference.

  The following applies here: First, the root directory of the project library is searched, then the subfolders.

Use the following XML structure to import an HMI tag of the "UDT" data type:

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
    ...
    <LinkList>
        <DataType TargetID="@OpenLink">
            <Name>HmiUdt_1 V 1.0.0</Name>    <- Must exist in the project library
        </DataType>
        ...
    </LinkList>
    ...
</Hmi.Tag.Tag>
```

6.3.3 VB scripts

6.3.3.1 Exporting VB scripts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

All sublevel user-defined folders are taken into account for the export. A separate XML file is created for each exported VB script.

Program code: Exporting a VB script

Modify the following program code to export a selected VB script of an HMI device to an XML file:

```
//Exports a single vbscript of an HMI device
private static void ExportSingleVBScriptOfHMITarget(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    VBScript vbScript = vbScripts.Find("MyVBScript");

    FileInfo info = new
FileInfo(string.Format(@"C:\OpennessSamples\Export\Scripts\{0}.xml", vbScript.Name));
    vbScript.Export(info, ExportOptions.None);
}
```

6.3.3.2 Exporting VB scripts from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

A separate XML file is created for each exported VB script.

Program code: Exporting a VB script from a user-defined folder

Modify the following program code to export a VB script from a user-defined folder to an XML file:

```
//Exports vbscripts of a selected vbscript system folder
private static void ExportVBScriptOfSelectedVBScriptSystemFolder(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbUserFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbUserFolder = vbUserFolders.Find("MyVBUserFolder");
    VBScriptComposition vbScripts = vbUserFolder.VBScripts;

    foreach (VBScript script in vbScripts)
    {
        FileInfo info = new
FileInfo(String.Format(@"C:\OpennessSamples\Export\Scripts\{0}.xml", script.Name));
        script.Export(info, ExportOptions.None);
    }
}
```

Program code: Exporting all VB scripts from a system folder

Modify the following program code to export all VB scripts from the system folder:

```
//Exports all vbscripts by using a foreach loop
private static void ExportAllVBScripts(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (vbScripts == null) return;

    foreach (VBScript script in vbScripts)
    {
        FileInfo info = new
FileInfo(string.Format(@"C:\OpennessSamples\Export\Scripts\{0}.xml", script.Name));
        script.Export(info, ExportOptions.None);
    }
}
```

6.3.3.3 Importing VB scripts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

Bulk imports are supported. As an alternative, you can use a program code with a Foreach loop (Exporting VB scripts (Page 1237)).

Program code

Modify the following program code to import a VB script from an XML file into an HMI device:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;

namespace ImportVBScripts
{
internal class Program
{
private static void Main(string[] args)
{
...
}
private static void ImportSingleVBScriptToHMITarget(HmiTarget hmitarget)
{
VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
if (vbScripts == null) return;
{
FileInfo info = new FileInfo(@"D:\Samples\Import\VBScript.xml");
vbScripts.Import(info, ImportOptions.None);
}
}
```

6.3.4 Text lists

6.3.4.1 Exporting text lists from an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

The export of text and graphic lists includes all their entries. Text and graphic lists can be exported separately.

The text lists of an HMI device are exported. A separate XML file is created for each exported text list.

Program code

Modify the following program code to export text lists from an HMI device:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;

namespace ExportTextListsFromHMIDevice
{
internal class Program
{
private static void Main(string[] args)
{
...
}
//Export TextLists
private static void ExportTextLists(HmiTarget hmitarget)
{
TextListComposition text = hmitarget.TextLists;
foreach (TextList textList in text)
{
FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml", textList.Name));
textList.Export(info, ExportOptions.WithDefaults);
}
}
}
}
```

6.3.4.2 Importing a text list into an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The API interface supports the import of a text list from an XML file into an HMI device.

Program code

Modify the following program code to import a text list from an XML file into an HMI device:

```
//Imports a single TextList
private static void ImportSingleTextList(HmiTarget hmitarget)
{
    TextListComposition textListComposition = hmitarget.TextLists;
    IList<TextList> importedTextLists = textListComposition.Import(new
FileInfo(@"D:\SamplesImport\myTextList.xml"), ImportOptions.Override);
}
```

6.3.4.3 Advanced XML formats for export/import of text lists

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- Standard export of text lists
  See Exporting text lists from an HMI device (Page 1240)
- Standard import of text lists
  See Importing text lists into an HMI device (Page 1242)

Application

A text list may also contain formatted texts. This primarily concerns the following formatting:

- Text formatting
- References to other objects within the text

Pure text formatting within a text list to be exported results in an advanced XML export format. Object references are characterized as Open Links. The same applies to text lists to be imported with formatted texts.

Advanced XML export formats may also become considerably more complex. For example, more than just the object name may be linked in the text list, perhaps by means of an Open Link to a PLC tag of a different device. In this case, all information must be coded in a string in order to remove the Open Link.

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
<!-- ... -->
    <MultilingualText ID="5" CompositionName="Text">
        <ObjectList>
            <MultilingualTextItem ID="6" CompositionName="Items">
                <AttributeList>
                    <Culture>en-US</Culture>
                    <Text>
                        <body>
                            <p>
                                <field ref="0" />
                            </p>
                        </body>
                        <fieldinfos>
                            <fieldinfo name="0" domaintype="HMICommonTextList">
                                <reference TargetID="@OpenLink">
                                    <name>Siemens.Simatic.Hmi.Utah.TextAndGraphicLists.HmiTextList:Empty Text_list_
                                </reference>
                                <subreference TargetID="@OpenLink">
                                    <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:t1</name>
                                </subreference>
                                <domaindata>
                                    <format length="9" />
                                </domaindata>
                            </fieldinfo>
                        </fieldinfos>
                    </Text>
                </AttributeList>
            </MultilingualTextItem>
            <MultilingualTextItem ID="7" CompositionName="Items">
                <AttributeList>
                    <Culture>de-CH</Culture>
                    <Text>
                        <body>
                            <p>
                                <field ref="0" />
                            </p>
                        </body>
                        <fieldinfos>
                            <fieldinfo name="0" domaintype="HMICommonTextList">
                                <reference TargetID="@OpenLink">
                                    <name>Siemens.Simatic.Hmi.Utah.TextAndGraphicLists.HmiTextList:Empty Text_list_
                                </reference>
                                <subreference TargetID="@OpenLink">
                                    <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:t1</name>
                                </subreference>
                                <domaindata>
                                    <format length="9" />
                                </domaindata>
                            </fieldinfo>
                        </fieldinfos>
                    </Text>
                </AttributeList>
            </MultilingualTextItem>
        </ObjectList>
    </MultilingualText>
```

6.3.5 Graphic lists

6.3.5.1 Exporting graphic lists

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The export of text and graphic lists includes all their entries. Text and graphic lists can be exported separately.

One XML file is created per graphic list. Global graphic objects contained in the graphic lists are exported as Open Links.

Program code

Modify the following program code to export graphic lists of an HMI device:

```
//Exports GraphicLists
private static void ExportGraphicLists(HmiTarget hmitarget)
{
    GraphicListComposition graphic = hmitarget.GraphicLists;
    foreach (GraphicList graphicList in graphic)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
graphicList.Name));
        graphicList.Export(info, ExportOptions.WithDefaults);
    }
}
```

6.3.5.2 Importing a graphic list

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The API interface supports the import of a graphic list from an XML file into an HMI device.

All referenced graphic objects of the graphic list are included in the import. References to global graphics are not included. If the referenced global graphics exist in the target project, the references to the global graphics are restored during the import.

Program code

Modify the following program code to import a graphic list from an XML file into an HMI device:

```
//Imports a single GraphicList
private static void ImportSingleGraphicList(HmiTarget hmitarget)
{
    GraphicListComposition graphicListComposition = hmitarget.GraphicLists;
    IList<GraphicList> importedGraphicLists = graphicListComposition.Import(new
FileInfo(@"D:\Samples\Import\myGraphicList.xml"), ImportOptions.Override);
}
```

6.3.6 Connections

6.3.6.1 Exporting connections

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The API interface supports the export of all connections of an HMI device to an XML file.

Note

Export of integrated connections

Export of integrated connections is not supported.

A separate XML file is created for each exported connection.

Program code

Modify the following program code to export all connections of an HMI device to an XML file:

```
//Exports communication connections from an HMI device
private static void ExportConnectionsFromHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    foreach(Connection connection in connections)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
connection.Name));
        connextion.Export(info, ExportOptions.WithDefaults);
    }
}
```

6.3.6.2 Importing connections

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The API interface supports the import of all connections of an HMI device from an XML file into an HMI device. If you want to import several communication connections, import the corresponding XML file for each one.

Note

If you import a connection into a project in which an integrated connection has already been configured, this connection is not overwritten. The import is canceled and an Exception is thrown.

Program code

Modify the following program code to import an individual connection of an HMI device from an XML file into an HMI device:

```
//Imports Communication connections to an HMI device
private static void ImportConnectionsToHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    IList<Connection> importedConnectionLists = connections.Import(new
FileInfo(@"D:\Samples\Import\myConnectionImport.xml"), ImportOptions.Override);
}
```

6.3.7 Screens

6.3.7.1 Overview of exportable screen objects

Application

You can export or import the screens below using TIA Portal Openness APIs:

Table 6-4 Supported screens

Object	Export/import possible
Screen	Yes
Global screen	Yes
Screen template	Yes
Permanent area	Yes
Pop-up screen	Yes
Slide-in screen	Yes

You can export or import the screen objects below using TIA Portal Openness APIs:

Table 6-5 Supported screen objects

Range	Object type	Export/import possible
Basic objects	Line	Yes
	Polyline	Yes
	Polygon	Yes
	Ellipse	Yes
	Ellipse segment	–
	Circle segment	–
	Elliptical arc	–
	Circular arc	–
	Circle	Yes
	Rectangle	Yes
	Connector	–
	Text field	Yes
	Graphic view	Yes
	Pipe	–
	Double T-piece	–
	T-piece	–
	Pipe elbow	–

Range	Object type	Export/import possible
Elements	I/O field	Yes
	Graphic I/O field	Yes
	Editable text field	–
	List box	–
	Combo box	–
	Button	Yes
	Round button	–
	Illuminated button	Yes
	Switch	Yes
	Symbolic I/O field	Yes
	Date/time field	Yes
	Bar	Yes
	Symbol library	Yes
	Slider	Yes
	Scroll bar	–
	Check box	–
	Option buttons	–
	Gauge	Yes
	Clock	Yes
	Memory space view	–
	Function keys (softkeys)	Yes
	Groups	Yes
	Faceplate instances	Yes

Range	Object type	Export/import possible
Controls	Screen window	–
	User view	Yes
	Print job/script diagnostics	–
	Camera view	–
	PDF view	–
	Recipe view	–
	Alarm view	–
	Alarm indicator	–
	Alarm window	–
	f(x) trend view	–
	f(t) trend view	–
	Table view	–
	Value table	–
	HTML Browser	–
	Media Player	–
	Channel diagnostics	–
	WLAN reception	–
	Zone name	–
	Zone signal	–
	Effective range name	–
	Effective range name (RFID)	–
	Effective range signal	–
	Charge condition	–
	Handwheel	–
	Help indicator	–
	Sm@rtClient view	–
	Status/Force	–
	Memory space view	–
	NC subprogram display	–
	System diagnostic view	–
	System diagnostic window	–

See also

Basic principles of importing/exporting (Page 1207)

6.3.7.2 Exporting all screens of an HMI device

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

A different program code is required for exporting all aggregated screens of all user-defined screen folders of an HMI device.

Program code: Exporting all screens of a device

Modify the following program code to export the screens of a user-defined screen folder of an HMI device and the screen system folder:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;

using Siemens.Engineering.SW.Blocks;

using Siemens.Engineering.SW.ExternalSources;

using Siemens.Engineering.SW.Tags;

using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;

namespace ExportAllScreensOfHMIDevice
{
internal class Program
{
private static void Main(string[] args)
{
...
}
private static void ExportAllScreensOfDevice(string rootPath, HmiTarget hmiTarget)
{
DirectoryInfo info = new DirectoryInfo(rootPath);
info.Create();
// Export the ScreenFolder recursive
string screenPath = Path.Combine(rootPath, "Screens");
info = new DirectoryInfo(screenPath);
info.Create();
ExportScreens(screenPath, hmiTarget);
}
// Export the screens of a user-defined screen folder of an HMI device and the screen system
folder
private static void ExportScreensOfDevice(HmiTarget hmiTarget)
{
ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("MyScreenFolder");
//or ScreenSystemFolder folder = hmiTarget.ScreenFolder;
ScreenComposition screens = folder.Screens;
foreach(Screen screen in screens)
{
```

```
FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml", folder.Name,
screen.Name));
screen.Export(info, ExportOptions.WithDefaults);
}
}
// Exporting all screens of a device independent of the user
public static void ExportScreens(string screenPath, HmiTarget target)
{
foreach(Screen screen in target.ScreenFolder.Screens)
{
screen.Export(new FileInfo(Path.Combine(screenPath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
}
foreach(ScreenUserFolder subfolder in target.ScreenFolder.Folders)
{
ExportScreenUserFolder(Path.Combine(screenPath, folder.Name), subfolder.Name);
}
}
private static void ExportScreenUserFolder(string screenPath,ScreenUserFolder folder )
{
foreach(Screen screen in folder.Screens)
{
screen.Export(new FileInfo(Path.Combine(screenPath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
}
foreach(ScreenUserFolder subfolder in folder.Folders)
{
ExportScreenUserFolder(Path.Combine(screenPath, subfolder.Name), subfolder);
}
}
}
}
```

6.3.7.3 Exporting a screen from a screen folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data of a screen is exported:

Screen	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Open links	Template
Compositions	• Layers • Animations All configured animations that are based on Runtime Advanced are exported. • Events All configured events that are based on Runtime Advanced are exported. • Softkeys All configured softkeys are exported.

The following data is exported for each layer:

Note

By default, the layer name in the TIA Portal is an empty text.

If you do not change the layer name in the TIA Portal, the exported layer name will be an empty text. In this case, the displayed layer name in the TIA Portal depends on the user interface language.

If you do change the layer name in the TIA Portal, the modified layer name is displayed in all relevant languages.

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems (with screen items)

Not included in the export:

- SCADA-specific attributes.
- Layers that do not contain any screen items and whose attributes do not differ from the default values.

Program code

Modify the following program code to export an individual screen from the user folder or from the system folder of an HMI device:

```
//Exports a single screen from a screen folder
private static void ExportSingleScreenFromScreenFolder(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find("Screen_1.xml");
    if (screen == null) return;
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

6.3.7.4 Importing screens to an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The screens can only be imported to a specific type of HMI device. The HMI device and the device from which the screens were exported must be of the same device type.

The following data of a screen is imported:

Screen	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Open links	Templates
Compositions	• Layers • Animations All animations configurable for screens are imported. • Events All events configurable for screens are imported. • Softkeys All softkeys configurable for screens are imported.

The following data is imported for each layer:

Note

If you have specified an empty text for the layer name before the import, the displayed layer name in the TIA Portal after the import depends on the user interface language.

If you have assigned a layer name, the specified layer name is displayed in all relevant languages after the import.

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems

Restrictions

- The import is canceled and an Exception is thrown if the width and height of a screen do not correspond to the dimensions of the device. Adaptation of the screen items contained is not supported. For this reason, certain screen items may be located beyond the screen boundaries. A compiler warning is output in this case.
- The screen number must be unique for all screens of the device. A screen import is canceled if a screen with a screen number that was already created in the device is found. If you have not yet assigned a screen number, a unique number is assigned to the screen during the import.
- The layout of the screen items within the Z-order must be unique and contiguous for each layer in the screen. For this reason, after the import of the screen, a consistency check is performed that repairs the layout, if necessary. This action may lead to modified "tab indexes" for certain screen items.
  You can change the Z-order of the screen items in the XML file manually. The screen item in first place is at the very end in the Z-order.

Note

You can change the values for width and height of the screen items in the XML file if the attribute "Fit size to content" is enabled for the screen item.

Note

Import of screen types from the library is not supported

As of WinCC V12 SP1, you can create a screen as type in the library. Instances of the screen type used in the project can be edited like other screens using the TIA Portal Openness application. When you export screens, the instances of screen types are exported without the type information.

When you re-import these screens into the project, the instances of the screen types are overwritten and the instance is detached from the screen type.

Program code: Importing screens to an HMI device

Modify the following program code to import screens to an HMI device using a For each loop:

```
//Imports all screens to an HMI device
private static void ImportScreensToHMITarget(HmiTarget hmitarget)
{
    FileInfo[] exportedScreens = new FileInfo[] {new
FileInfo(@"D:\Samples\Import\Screen_1.xml"), new
FileInfo(@"D:\Samples\Import\Screen_2.xml")};
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    foreach (FileInfo screenFileInfo in exportedScreens)
    {
        folder.Screens.Import(screenFileInfo, ImportOptions.Override);
    }
}
```

Program code: Import to a newly created user folder

Modify the following program code to import a screen to a newly created user folder of an HMI device:

```
//Imports a single screen to a new created user folder of an HMI device
private static void ImportSingleScreenToNewFolderOfHMITarget(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Create("MyFolder");
    folder.Screens.Import(new FileInfo(@"D:\Samples\Import\myScreens.xml"),
ImportOptions.Override);
}
```

6.3.7.5 Exporting permanent areas

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data of the permanent area is exported:

Permanent area	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name
Compositions	Layers

The following data is exported for each layer:

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems (with screen items)

Program code

Modify the following program code to export a permanent area of an HMI device to an XML file:

```
//Exports a permanent area
private static void ExportScreenoverview(HmiTarget hmitarget)
{
    ScreenOverview overview = hmitarget.ScreenOverview;
    if (overview == null) return;

    FileInfo info = new FileInfo(@"D:\Samples\Screens\ExportedOverview.xml");
    overview.Export(info, ExportOptions.WithDefaults);
}
```

6.3.7.6 Importing permanent areas

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data of the permanent area is imported:

Permanent area	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, Visible, Number
Compositions	Layers

The following data is imported for each layer:

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems (with screen items)

The import is canceled and an Exception is thrown if the width and height of a screen do not correspond to the dimensions of the device. Adaptation of the included device items (Screen items) is not supported. For this reason, some device items may be located beyond the screen boundaries. A compiler warning is output in this case.

The layout of the device items must be unique and contiguous in the permanent area. For this reason, after the import of the permanent area, a consistency check is performed that repairs the layout, if necessary. This action may lead to modified "tab indexes" for certain device items.

Program code

Modify the following program code to import a permanent area from an XML file into an HMI device:

```
//Imports a permanent area
private static void ImportScreenOverview(HmiTarget hmiTarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\ExportedOverview.xml");
    hmiTarget.ImportScreenOverview(info, ImportOptions.Override);
}
```

6.3.7.7 Exporting all screen templates of an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

One XML file is created per screen template.

Because bulk exports are not supported, you need to enumerate and export all screen templates separately. In the course of this action, make sure that the screen template names used conform to the file naming conventions of your file system.

Program code: Exporting all screen templates of a device

Modify the following program code to export all screen templates from a specific folder:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;

namespace ExportAllScreenTemplatesOfHMIDevice
{
internal class Program
{
private static void Main(string[] args)
{
...
}
public static void ExportScreenTemplatesOfDevice(string rootPath, ScreenTemplateUserFolder
folder)
{
string screenPath = Path.Combine(rootPath, "Screens");
DirectoryInfo info = new DirectoryInfo(screenPath);
info.Create();
//Export the ScreenTemplateFolder recursive
ExportScreenTemplates (screenPath, hmiTarget);
}
//Exports all screen templates of a selected folder
private static void ExportScreenTemplates(string templatePath, HmiTarget hmiTarget)
{
foreach (ScreenTemplate screen in hmiTarget.ScreenTemplateFolder.ScreenTemplates)
{
screen.Export(new FileInfo(Path.Combine(templatePath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
}
foreach (ScreenTemplateUserFolder folder in hmiTarget.ScreenTemplateFolder.Folders)
{
ExportScreenTemplates(Path.Combine(templatePath, folder.Name), hmiTarget);
}
}
}
```

```
}
```

6.3.7.8 Exporting screen templates from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data of the screen template is exported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name
Compositions	• Layers • Animations All configured animations are exported. SCADA animations are not exported. • Softkeys All configured softkeys are exported.

The following data is exported for each layer:

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems (with screen items)

Program code: Exporting a screen template of a user-defined folder

Modify the following program code to export an individual screen template from the system folder or from a user-defined folder:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;

namespace ExportScreenTemplatesOfFolder
{
internal class Program
{
private static void Main(string[] args)
{
...
}
private static void ExportSingleScreenTemplate(string templatePath, HmiTarget hmiTarget)
{
ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
//or ScreenTemplateSystemFolder folder = hmiTarget.ScreenTemplateFolder;
ScreenTemplateComposition templates = folder.ScreenTemplates;
ScreenTemplate template = templates.Find("templateName");
if(template == null) return;
FileInfo info = new FileInfo(string.Format(@"D:\Samples\Templates\{0}\{1}.xml",
folder.Name, template.Name));
template.Export(info, ExportOptions.WithDefaults);
}
// Exporting all screen templates of a user-defined folder
public static void ExportScreenTemplateUserFolder(string rootPath,
ScreenTemplateUserFolder folder)
{
DirectoryInfo info = new DirectoryInfo(rootPath);
info.Create();
foreach (ScreenTemplate screen in folder.ScreenTemplates)
{
screen.Export(new FileInfo(Path.Combine(info.FullName, screen.Name + ".xml")),
ExportOptions.WithDefaults);
}
```

```
foreach (ScreenTemplateUserFolder subfolder in folder.Folders)
{
ExportScreenTemplateUserFolder(Path.Combine(info.FullName, subfolder.Name), subfolder);
}
}
// Exports all screen templates of a selected folder
private static void ExportScreenTemplates(string templatePath, ScreenTemplateUserFolder
folder)
{
foreach (ScreenTemplate screen in folder.ScreenTemplates)
{
screen.Export(new FileInfo(Path.Combine(templatePath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
}
foreach (ScreenTemplateUserFolder subfolder in folder.Folders)
{
ExportScreenTemplates(Path.Combine(templatePath, subfolder.Name), subfolder);
}
}
}
}
```

6.3.7.9 Importing screen templates

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

The following data of a screen template is imported:

Screen template	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, SetTabOrderInFront
Compositions	• Layers • Animations All animations configurable for screens are imported. • Softkeys All softkeys configurable for screens are imported.

The following data is imported for each layer:

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems (with screen items)

The import is canceled and an Exception is thrown if the width and height of a screen template do not correspond to the dimensions of the device. Adaptation of the included screen items is not supported. For this reason, certain screen items may be located beyond the screen boundaries. A compiler warning is output in this case.

The layout of the screen items must be unique and contiguous in the screen template. For this reason, after the import of the screen template, a consistency check is performed which repairs the layout, if necessary. This action may lead to modified "tab indexes" for certain screen items.

Program code: General import

Modify the following program code to import all screen templates to an HMI device using a For each loop:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;

namespace ImportingScreenTemplates
{
internal class Program
{
//Imports screen templates to an HMI device
private static void ImportScreenTemplatesToHMITarget(HmiTarget hmiTarget)
{
ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
// or ScreenTemplateSystemFolder folder = hmiTarget.ScreenTemplateFolder;
FileInfo[] exportedTemplates = new FileInfo[] {new
FileInfo(@"D:\Samples\Import\Template_1.xml"), new
FileInfo(@"D:\Samples\Import\Template_n.xml")};
foreach (FileInfo templateFileName in exportedTemplates)
{
folder.ScreenTemplates.Import(templateFileName, ImportOptions.Override);
}
}
//Imports screen templates to a user folder of an HMI device
private static void ImportScreenTemplatesToFolderOfHMITarget(HmiTarget hmiTarget)
{
ScreenTemplateUserFolder screenTemplateFolder =
hmiTarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
ScreenTemplateUserFolder folder = screenTemplateFolder.Folders.Create("MyNewFolder");
folder.ScreenTemplates.Import(new FileInfo(@"D:\Samples\Import\ScreenTemplate.xml"),
ImportOptions.Override);
}
}
}
```

6.3.7.10 Exporting a pop-up screen

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

The following data of the pop-up screen is exported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
Compositions	• Layers • Events All configured events are exported.

The following data is exported for each layer:

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems All exportabe screen objects are exported.

Program code: Exporting a pop-up screen from a folder

Modify the following program code to export an individual pop-up screen from the system folder or from a user-defined folder:

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;

namespace ExportingAPopupImage
{
internal class Program
{
//Exports a single pop-up screen
private static void ExportSinglePopUpScreen(HmiTarget hmitarget)
{
ScreenPopupUserFolder folder = hmitarget.ScreenPopupFolder.Folders.Find("MyPopupFolder");
//or ScreenPopupSystemFolder folder = hmitarget.ScreenPopupFolder;
ScreenPopupComposition popups = folder.ScreenPopups;
ScreenPopup popup = popups.Find("popupName");
if(popup == null) return;
FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml", folder.Name,
popup.Name));
popup.Export(info, ExportOptions.WithDefaults);
}
}
}
```

6.3.7.11 Importing a pop-up screen

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data of a pop-up screen is imported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
Compositions	• Layers • Events All configured events are exported.

The existence of the following attributes is mandatory for the import:

- Name
- Height
- Width

The following data is imported for each layer:

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems All importable screen objects are imported.

Restrictions

If a device doesn't support pop-up screens, the import is cancelled and an Exception is thrown.

If the width and height of a pop-up screen do not comply to the following dimensions restrictions for a device, the import is cancelled and an Exception is thrown:

- Minimum height = 1 pixel
- Minimum width = 1 pixel
- Maximum height = sixfold height of the device's screen
- Maximum width = twofold width of the device's screen
- For devices with runtime version V13 SP1 the maximum height and the maximum width is equal with th height and width of the device's screen.

Program code: Importing a pop-up screen into a folder

Modify the following program code to import a pop-up screen into the pop-up screen system folder or to a user-defined folder:

```
//Imports a pop-up screen to an HMI device
private static void ImportPopupScreenToHMITarget(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\PopupScreen.xml"));
    hmitarget.ScreenPopupFolder.ScreenPopups.Import(info, ImportOptions.None);
}
```

6.3.7.12 Exporting a slide-in screen

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data and values of the slide-in screen is exported:

Screen templates	Data	
Attributes	Activate	false
	ActiveLayer	0
	BackColor	(182; 182; 182)
	GridColor	(0; 0; 0)
	Dimension	427 The attribute "Dimension" specifies either the width or height of the slide-in screen, depending on the used slide-in screen type.
	LineColor1	(223; 223; 223)
	LineColor2	(32; 32; 32)
	OperatableAreaColor	(128; 128; 128)
	SlideinType	Top, Bottom, Left, Right Slide-in screens do not have a name but a SlideinType.
	Visibility	FadeOut
Compositions	Layers	

Note

Slide-in screens do not have a name but a SlideinType.

The following data is exported for each layer:

Layer	Data	
Áttributes	Name,	
	Index	
	VisibleES	
Compositions	ScreenItems	All exportabe screen objects are exported.

Program code: Exporting a slide-in screen

Modify the following program code to export an individual slide-in screen from the system folder:

```
//Exports a single slide-in screen
private static void ExportSingleSlideinScreen(HmiTarget hmitarget)
{
    ScreenSlideinSystemFolder systemFolder = hmitarget.ScreenSlideinFolder;
    var screens = systemFolder.ScreenSlideins;
    ScreenSlidein slidein = screens.Find(SlideinType.Bottom);
    if (slidein == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml"));
    slidein.Export(info, ExportOptions.WithDefaults);
}
```

6.3.7.13 Importing a slide-in screen

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data and values of a slide-in screen is imported:

Screen templates	Data
Attributes	Activate = false ActiveLayer = 0 Authorization BackColor = (182; 182; 182) Dimension = 427 The attribute "Dimension" specifies either the width or height of the slide-in screen, depending on which of the two attributes is modifiable for the specified slide-in type. GridColor = (0; 0; 0) LineColor1 = (223; 223; 223) LineColor2 = (32; 32; 32) OperateableAreaColor = (128; 128; 128) SlideinType = Top, Bottom, Left, Right Visibility = FadeOut
Compositions	Layers

The existence of the following attribute is mandatory for the import:

- SlideinType

The following data is imported for each layer:

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems All importable screen objects are imported.

Restrictions

- If a device does not support slide-in screens, the import is cancelled and an Exception is thrown.
- If a slide-in screen is referenced from another element, the slide-in screen must be referenced via openlink and not via SlideinType, e. g. in system function "ShowSlideinScreen").
  The following table shows the mapping of the attribute "SlideinType" with the corresponding openlink:

SlideinType	Openlink Name
Top	GraphX_Slidein_Top
Right	GraphX_Slidein_Right
Bottom	GraphX_Slidein_Bottom
Left	GraphX_Slidein_Left

Program code: Importing a slide-in screen into a folder

Modify the following program code to import a slide-in screen to the slide-in screen system folder:

```
//Imports a slide-in screen to an HMI device
private static void ImportSlideinScreenToHMITarget(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\SlideInScreen.xml");
    hmitarget.ScreenSlideinFolder.ScreenSlideins.Import(info, ImportOptions.None);
}
```

6.3.7.14 Exporting a screen with a faceplate instance

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data of a faceplate instance in a screen is exported:

Screen	Data
Attributes	Left, Top, Width, Height, ObjectName, Resizing, TabIndex, FaceplateTypeName
Interface Attributes	All configured interface attributes of a faceplate instance are exported for exportable screen items.
Compositions	• Animations All movement animations are exported. Tag animations rely on interface attributes. • Events All configured events are exported.

Regard the following specifications for exported attributes of faceplate instance:

- Resizing
  The attribute "Resizing" is exported in any case, independent of the export options.
- FaceplateTypeName
  The attribute "FaceplateTypeName" identifies the corresponding faceplate type and version, e. g. "Faceplate_1 V 0.0.2".

Note

Faceplate type in a library folder

If a faceplate type is located within a library folder, the complete path and name is required to identify the faceplate type. The keyword "@$@" is used to separate folders and/or faceplate type name, e. g. "Folder_1@$@SubFolder_1@$@Faceplate_1 V 0.0.2".

The following data of inner screen items of a faceplate instance is excluded from the export:

Screen item	Attribute
IO-Field	Flashing on limit violation
Graphic IO-Field	Fit embedded graphic object to screen size

Program code

Modify the following program code to export an individual screen including a faceplate instance:

```
//Exports a single screen including a faceplate instance
private static void ExportSingleScreenWithFaceplateInstance(HmiTarget hmitarget)
{
    ScreenFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find("ScreenWithFaceplateName");
    if (screen == null) return;
    {
       FileInfo info = new FileInfo(string.Format(@"D:\Samples\Faceplates\{0}\{1}.xml",
folder.Name, screen.Name));
       screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

6.3.7.15 Importing a screen with a faceplate instance

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

The following data of a faceplate instance in a screen is imported:

Screen	Data
Attributes	Left, Top, Width, Height, ObjectName, Resizing, TabIndex, FaceplateTypeName
Interface Attributes	All configured interface attributes of a faceplate instance are imported for importable screen items.
Compositions	• Animations All movement animations are imported. Tag animations rely on interface attributes. • Events All configured events are imported.

The existence of the following attributes is mandatory for the import:

- ObjectName
- FaceplateTypeName

The following data of inner screen items of a faceplate instance is excluded from the export and import:

Screen item	**Attribute**
IO-Field	Flashing on limit violation
Graphic IO-Field	Fit embedded graphic object to screen size

Restrictions

- Unknown Faceplate, event or interface attribute
  If a faceplate type name, an event name or an interface attribute name is specified in the import file which does not exist in the project, the import is aborted with an Exception.
- Resizing behavior of a faceplate instance
  The attribute "Resizing" is imported in any case, independent of the export options.
  Examples:
  If "Resizing" is set to "KeepRatio", the "Height" attribute is used to calculate the "Width" attribute value.
  - The size of a faceplate type is 100 x 100 pixel. If a faceplate instance is imported with size 300 x 100 pixel and value "FixedSize" is set for the "Resizing" attribute, the import succeeds and the faceplate size is set to 100 x 100 pixel.
  - The size of a faceplate type is 100 x 50 pixel. A faceplate instance is imported with size 100 x 100 pixel and value "KeepRatio" is set for the "Resizing" attribute.The import succeeds and the faceplate size is set to 200 x 100 pixel.

  Note

  Sizing behavior of imported faceplate instances

  The values of "Resizing" and values of interface attributes can affect the size of the imported faceplate instance and even the size of the inclosed screen items.

  To avoid unrequested changes of the appearance of a faceplate instance, import a faceplate with the initial size or even without "Width" and "Height" attribute values.

- Deviant interface attribute values
  - If you modify attributes for the import, the last applied interface attribute value is imported.
  - If attributes depend on each other, other attribute values can be changed during the import.
    Example: A faceplate includes an I/O field. The attribute "Mode" is connected to an interface attribute. If you first set the mode to "Output" and then set the attribute "Hidden input" to true, the value of "Hidden input" is not applied after import. The first modification set the attribute "Hidden input" to read-only and therefore the value cannot be applied.
  - If a attribute value does not fulfill the restrictions of WinCC, the faceplates type value is displayed.
    Example: The display range of a gauge is set from 10 - 80. The attributes "MaximumValue" and "MinimumValue" are configured at interface attributes. If you set a minimum value that exceeds the maximum value, e. g. 100, the faceplate type's value for "MinimumValue" is displayed after import.
  - If an interface attribute is connected with several screen item attributes within the faceplate type, the interface attribute value at the faceplate instance will display the applied attribute value of the first connected screen item.
    Example: A faceplate includes two gauge objects with deviant maximum values. The minimum values of both gauges are connected to one single interface attribute.
    If you first set a minimum value that is applicable for both gauges, both values are set.
    If you set than a value that is only applicable for the second gauge, the value is only set for the second gauge, but the value of the first gauge is displayed as interface attribute.

Program code: Importing screens including a faceplate instance

Modify the following program code to import a screen including a faceplate instance:

```
//Imports single screen including a faceplate instance
private static void ImportSingleScreenWithFaceplateInstance(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\ScreenFaceplate.xml");
    hmitarget.ScreenFolder.Screens.Import(info, ImportOptions.None);
}
```

6.4 Importing/exporting data of a PLC device

6.4.1 CFC Charts (Export/Import)

6.4.1.1 Export/Import of CFC charts

The TIA Portal Openness API supports the export and import of charts that are created with STEP 7 CFC ("Continuous Function Chart").

You can export and import all charts as XML files, or export only selected charts.

You call these functions for defined tasks outside the TIA Portal by means of the Public API.

Note

Recommended PC hardware

If you are working with large projects, check that the computer meets the TIA Portal hardware requirements:

- RAM:
  32 GB for large projects

Functions

The following functions are available for CFC charts:

- Exporting CFC charts (Page 1284)
- Exporting only selected CFC charts (Page 1286)

- Importing CFC charts (Page 1287)
- Configuring chart passwords:
  - Setting a password for CFC charts (Page 1288)
  - Reading the password from CFC charts (Page 1290)
  - Changing a password for CFC charts (Page 1291)
  - Removing a password from CFC charts (Page 1293)

The function descriptions contain code samples that you can adapt to your openness program.

Security measures for TIA Portal Openness applications

It is recommended

- to install a TIA Portal Openness application with admin rights to the programs folder.
- to avoid the dynamical loading of program parts like assemblies or dlls from the users area.
- to run the TIA Portal Openness application with user rights.

Note

Siemens is not liable for and does not guarantee the compatibility of the data and information transported via these interfaces with third-party software.

We expressly point out that improper use of the interfaces can result in data loss or production downtimes.

Note

There are no obligations or guarantees of any kind associated with using this description to manually modify and evaluate the source file.

Siemens therefore accepts no liability arising from the use of all or part of this description.

If you import externally created configuration data that contains code errors, a wrong structure or unwanted manipulations, this can cause unexpected errors and security risks.

WARNING

The API user is responsible for ensuring the security measures of handling passwords through code.

More information in the TIA Portal Openness documentation:

- "Readme TIA Portal Openness"
- "Basics > Openness tasks > Introduction (Page 48)"
- "TIA Portal Openness API > General functions > TIA Portal Openness firewall (Page 88)"

- "TIA Portal Openness API > Functions for accessing the data of a PLC device > Functions for downloading data to PLC device > Downloading to PLC devices (Page 385)"
- "Export/import > Overview > Basic principles of importing/exporting (Page 1207)"

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

TIA Portal project view: Exporting and importing CFC charts (Page 1283)

6.4.1.2 TIA Portal project view: Exporting and importing CFC charts

You can export and import CFC charts in the TIA Portal project via the "Export / Import CFC" dialog.

The dialog uses the Openness functions for exporting and importing CFC charts. The same constraints apply as when using the Openness functions directly.

Follow the instructions under "Export/Import of CFC charts (Page 1281)", section "Security measures for TIA Portal Openness applications".

Data transfer dialog

For the import, you use the "Data transfer - Generate/import" dialog.

More information on this dialog:

- Industry Online Support: "SIMATIC Process Control System PCS 7 Help on data transfer dialog" (https://support.industry.siemens.com/cs/ww/en/view/109812471)

Requirements

- "TIA Openness" is installed.
- The PLC is not online.
- The CFC charts to be exported are not password-protected.
  CFC charts with configured passwords are ignored during the export.
- Import: The imported block types have been created and compiled under the PLC in the TIA Portal.
  If the imported XML file contains blocks that have not yet been created, the import is canceled.

Procedure

1. Select the "Charts" entry or a CFC chart in the project tree.
2. Select the "Tools > Import / Export CFC" entry in the menu bar.
   The "Export / Import CFC" dialog opens.

3. Select the action you want:
   - Export all CFC charts of the PLC
     The block types and the settings for tasks and run sequence are also exported along with the CFC charts.
   - Export individual CFC charts
     The CFC charts are exported without block types, task settings, and run sequence.
     Click "Next" and select the desired charts.
   - Import CFC charts
     You can select the scope of the import in the next "Data transfer - Generate/import" dialog.
4. Click "Next".
5. Select the storage path of the XML file and click "Finish".
   The export or import is started.
6. If you are exporting CFC charts, confirm the message that the XML file has been written.
   The XML file with the exported data is located in the selected storage path.
7. If you are importing CFC charts, the "Data transfer - Generate/import" dialog opens.
   Select the objects to be imported, and click the "Import objects from B to A" button in the "Generate/import" area: 
   The desired data is imported into the TIA Portal project under the PLC.
   The Inspector window contains information on the import where appropriate.

See also

Exporting CFC charts (Page 1284)

Exporting only selected CFC charts (Page 1286)

Importing CFC charts (Page 1287)

6.4.1.3 Exporting CFC charts

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See "Connecting to the TIA Portal (Page 82)"
- A project is open.
  See "Opening a project (Page 128)"
- PLC is not online.
- The CFC charts to be exported are not password-protected.
  CFC charts with a configured password are ignored during export.

Application

The TIA Portal Openness API supports the export of CFC charts to an XML file with the function "CompleteExport".

The function writes the CFC project data from the chart folder into an XML file:

- All CFC charts created under the selected PLC
- Block types used in the exported charts
- Task assignment for the exported charts
- Run sequence of each exported chart

You find more information on the supported objects in the CFC documentation: "Instructions and blocks".

If you want to start a selective XML export of CFC charts, use the function "SelectiveExport (Page 1286)".

Parameters

Parameter	Data type	Description
xmlFilePath	String	Folder path and name of the import file
modelVersion	String	S7TIA exchange model version to be used
filter	Int64	Filter options for the automation interface
unattended	Boolean	Toggle for silent mode

Program code

Modify the following program code to export all CFC charts from a PLC with their objects to an XML file.

```
plcSoftware = (PlcSoftware) swContainer.Software;

chartProvider = plcSoftware.GetService<ChartProviderS7>();
if (chartProvider == null)    // in case that CFC is not installed
    return;

// Export CFC charts:
// XML file information for export
    chartProvider.CompleteExport(@"D:\Users\username1\Documents\Automation\OpennessExample.
xml", "V2.0", 0, true);
```

More information in the TIA Portal Openness documentation:

- "Export/import > Overview > Exporting configuration data (Page 1211)"

See also

Export/Import of CFC charts (Page 1281)

TIA Portal project view: Exporting and importing CFC charts (Page 1283)

6.4.1.4 Exporting only selected CFC charts

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See "Connecting to the TIA Portal (Page 82)"
- A project is open.
  See "Opening a project (Page 128)"
- PLC is not online.
- The CFC charts to be exported are not password-protected.
  CFC charts with a configured password are ignored during export.

Application

The TIA Portal Openness API supports the export of specific CFC charts to an XML file with the function "SelectiveExport".

The function writes the project data for only the selected charts into an XML file:

- Specific CFC charts from the selected PLC
- Block types used in the exported charts
- Task assignment for the exported charts
- Run sequence of each exported chart

You find more information on the supported objects in the CFC documentation: "Instructions and blocks".

If you want to start a complete XML export of all CFC charts, use the function "CompleteExport (Page 1284)".

Parameters

Parameter	Data type	Description
xmlFilePath	String	Folder path and name of the import file
selectedObjects	String[]	Names of the charts that you want to export
modelVersion	String	S7TIA exchange model version to be used
filter	Int64	Filter options for the automation interface
unattended	Boolean	Toggle for silent mode

Program code

Modify the following program code to export only selected CFC charts with their objects to an XML file.

```
plcSoftware = (PlcSoftware) swContainer.Software;

chartProvider = plcSoftware.GetService<ChartProviderS7>();
if (chartProvider == null)    // in case that CFC is not installed
    return;

// Export selected CFC charts
// XML file information for export of CFC charts "CFC_1" and "CFC_3"
    chartProvider.SelectiveExport(@"D:\Users\username1\Documents\Automation\OpennessExample
.xml", new string[] { "CFC_1", "CFC_3" }, "V2.0", 0, true);
```

More information in the TIA Portal Openness documentation:

- "Export/import > Overview > Exporting configuration data (Page 1211)"

See also

Export/Import of CFC charts (Page 1281)

TIA Portal project view: Exporting and importing CFC charts (Page 1283)

6.4.1.5 Importing CFC charts

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See "Connecting to the TIA Portal (Page 82)"
- A project is open.
  See "Opening a project (Page 128)"
- PLC is not online.

Application

The TIA Portal Openness API supports the import of CFC charts from an XML file with the function "Import".

The XML file is created by a complete XML export or by a selective XML export of specific CFC charts.

Parameters

Parameter	Data type	Description
xmlFilePath	String	Folder path and name of the import file
modelVersion	String	S7TIA exchange model version to be used

Parameter	Data type	Description
filter	Int64	Filter options for the automation interface In the current CFC version, the parameter is not evaluated during export and import and has no function.
unattended	Boolean	Toggle for silent mode
deleteAtTarget	Boolean	Toggle for deleting objects in the TIA project that were not included in the original export file

Program code

Modify the following program code to import CFC charts from an XML file.

```
plcSoftware = (PlcSoftware) swContainer.Software;

chartProvider = plcSoftware.GetService<ChartProviderS7>();
if (chartProvider == null)    // in case that CFC is not installed
    return;

// Import CFC charts
// XML file information for import
    chartProvider.Import(@"D:\Users\username1\Documents\Automation\OpennessExample.xml",
"V2.0", 0, true, false);
```

More information in the TIA Portal Openness documentation:

- "Export/import > Overview > Importing configuration data (Page 1212)"

See also

Export/Import of CFC charts (Page 1281)

TIA Portal project view: Exporting and importing CFC charts (Page 1283)

6.4.1.6 Setting a password for CFC charts

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See "Connecting to the TIA Portal (Page 82)"
- A project is open.
  See "Opening a project (Page 128)"
- PLC is not online.

Application

To protect a CFC chart or hierarchical CFC chart from unintentional editing, you can protect the chart with a password.

- To configure a password, use the function "AddChartProtection".
- To change an existing password, use the function "ChangeChartProtection (Page 1291)".
- To read the password from a chart as a hash value, use the function "GetChartProtection (Page 1290)".
- To delete the configured password, use the function "RemoveChartProtection (Page 1293)".

NOTICE
Password: No authorization, no know-how protection of the charts The password merely protects the CFC chart from unintentional editing. This type of access protection is not intended to increase the access security. The password does not offer: • Protection against unauthorized access to know-how in CFC charts • Security-relevant authorization for access to CFC charts

Parameters

Parameter	Data type	Description
chartName	System.String	Name of the chart which is protected with a password.
newHashedPassword	System.String	Password The password is displayed as a hash value.

Return value

The function "AddChartProtection" returns a System.Boolean:

TRUE	The password was set successfully.
FALSE	The password could not be set.

Program code

Modify the following program code to set a password for a CFC chart.

In this example, the hash value of the new password is included as an abbreviated example value.

```
plcSoftware = (PlcSoftware) swContainer.Software;

chartProvider = plcSoftware.GetService<ChartProviderS7>();
if (chartProvider == null)    // in case that CFC is not installed
    return;

// Configure CFC chart password
    bool added = chartProvider.AddChartProtection("CFC_1", "AgGUWq...92M=");
```

See also

Export/Import of CFC charts (Page 1281)

6.4.1.7 Reading the password from CFC charts

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See "Connecting to the TIA Portal (Page 82)"
- A project is open.
  See "Opening a project (Page 128)"
- PLC is not online.

Application

To read the password from a CFC chart, use the function "GetChartProtection".

You can change or delete that password with the functions "ChangeChartProtection (Page 1291)" and "RemoveChartProtection (Page 1293)".

NOTICE
Password: No authorization, no know-how protection of the charts The password merely protects the CFC chart from unintentional editing. This type of access protection is not intended to increase the access security. The password does not offer: • Protection against unauthorized access to know-how in CFC charts • Security-relevant authorization for access to CFC charts

Parameters

Parameter	Data type	Description
chartName	System.String	Name of the chart which is protected with a password.

Return value

The function "GetChartProtection" returns a System.Boolean:

TRUE	The password was read successfully.
FALSE	The password could not be read.

Program code

Modify the following program code to read a password from a CFC chart.

```
plcSoftware = (PlcSoftware) swContainer.Software;

chartProvider = plcSoftware.GetService<ChartProviderS7>();
if (chartProvider == null)    // in case that CFC is not installed
    return;

// Read CFC chart password
    string passwordHash = chartProvider.GetChartProtection("CFC_1");
```

See also

Setting a password for CFC charts (Page 1288)

Export/Import of CFC charts (Page 1281)

6.4.1.8 Changing a password for CFC charts

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See "Connecting to the TIA Portal (Page 82)"
- A project is open.
  See "Opening a project (Page 128)"
- PLC is not online.

Application

To change the password that is used to protect a CFC chart or hierarchical CFC chart from unintentional editing, use the function "ChangeChartProtection".

You can read or delete a password with the functions "GetChartProtection (Page 1290)" and "RemoveChartProtection (Page 1293)".

NOTICE
Password: No authorization, no know-how protection of the charts The password merely protects the CFC chart from unintentional editing. This type of access protection is not intended to increase the access security. The password does not offer: • Protection against unauthorized access to know-how in CFC charts • Security-relevant authorization for access to CFC charts

Parameters

Parameter	Data type	Description
chartName	System.String	Name of the chart which is protected with a password.
currentPassword	System.Security.SecureString	Currently used password for the CFC chart.
newHashedPassword	System.String	New Password The password is displayed as a hash value.

Return value

The function "ChangeChartProtection" returns a System.Boolean:

TRUE	The password was changed successfully.
FALSE	The password could not be changed.

Program code

Modify the following program code to change a password for a CFC chart.

In this example, the password "test" is changed to a new password. The hash value of the new password is included as an abbreviated example value.

```
plcSoftware = (PlcSoftware) swContainer.Software;

chartProvider = plcSoftware.GetService<ChartProviderS7>();
if (chartProvider == null)    // in case that CFC is not installed
    return;

// Change CFC chart password
char[] password1 = new char[] {'t', 'e', 's', 't'};

SecureString securePassword1 = new SecureString();
foreach (char ch in password1)
    securePassword1.AppendChar(ch);

bool changed = chartProvider.ChangeChartProtection("CFC_1", securePassword1,
"AgGUWq...92M=");
```

See also

Setting a password for CFC charts (Page 1288)

Export/Import of CFC charts (Page 1281)

6.4.1.9 Removing a password from CFC charts

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See "Connecting to the TIA Portal (Page 82)"
- A project is open.
  See "Opening a project (Page 128)"
- PLC is not online.

Application

To delete a configured password for a CFC chart, use the function "RemoveChartProtection".

The CFC chart can then be opened and edited again without entering a password.

To set the password again, use the function "AddChartProtection (Page 1288)".

NOTICE
Password: No authorization, no know-how protection of the charts The password merely protects the CFC chart from unintentional editing. This type of access protection is not intended to increase the access security. The password does not offer: • Protection against unauthorized access to know-how in CFC charts • Security-relevant authorization for access to CFC charts

Parameters

Parameter	Data type	Description
chartName	System.String	Name of the chart which is protected with a password.
currentPassword	System.Security.SecureString	Currently used password for the CFC chart.

Return value

The function "RemoveChartProtection" returns a System.Boolean:

TRUE	The password was deleted successfully.
FALSE	The password could not be deleted.

Program code

Modify the following program code to remove a password for a CFC chart.

In this example, the password "test" was configured for the chart "CFC_1".

```
plcSoftware = (PlcSoftware) swContainer.Software;

chartProvider = plcSoftware.GetService<ChartProviderS7>();
if (chartProvider == null)    // in case that CFC is not installed
    return;

// Remove CFC chart password
char[] password1 = new char[] {'t', 'e', 's', 't'};

SecureString securePassword1 = new SecureString();
foreach (char ch in password1)
    securePassword1.AppendChar(ch);

bool removed = chartProvider.RemoveChartProtection("CFC_1", securePassword1);
```

See also

Changing a password for CFC charts (Page 1291)

Reading the password from CFC charts (Page 1290)

Export/Import of CFC charts (Page 1281)

6.4.2 Blocks

6.4.2.1 XML structure of the block interface section

Basic principle

The data in the export file from the import/export is structured with reference to a basic structure. Every import file has to fulfill the basic structural conditions.

The export file includes all edited tags and constants of the interface section of an exported block. All attributes with "`ReadOnly`="TRUE""and "`Informative`="TRUE"" are excluded.

If the information is redundant, it must exactly be identical in the import XML file and the project data. Otherwise the import will throw a recoverable exception.

The project data can contain more data than the import XML file, e. g. an external type may have additional members

Only writeable values can be imported via TIA Portal Openness XML.

Depending on the TIA Portal Openness export settings, the export file includes a defined set of attributes and elements. The XML exported from higher versions of the product is not compatible during the import operation in lower version of the TIA Portal.

Basic structure

The interface section of an exported block is covered in the <Interface> element in the SimaticML of a block. The root object is the <Sections> element, which represents the interface section of an exported block. The sequence of the following description of elements represents the required sequence in the input file.

```
<Interface>
    <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v1">
        <Section Name="Input">
            <Member Name="input1" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
              <AttributeList>
                ...
              </AttributeList>
            </Member>
            <Member Name="input2" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
                <AttributeList>
                  ...
                </AttributeList>
            </Member>
        </Section>
        <Section Name="Output">
            <Member Name="output1" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
                <AttributeList>
                  ...
                </AttributeList>
            </Member>
        </Section>
        <Section Name="InOut" />
        <Section Name="Static" />
        <Section Name="Temp" />
        <Section Name="Constant" />
    </Sections>
</Interface>
```

- Section
  Section represents a single parameter or local data of a program block
- Member
  Member represents the tags or constants used in the program block. Depending of the datatype of a tag, members can be nested or have further structural sub elements.
  In case of the data type "ARRAY" the structural element "Subelement Path" represents e. g. the index of the components of an array element.
  Only those members are exported, which were edited by the user.
- AttributeList
  The <AttributeList> includes all defined attributes of a member. Attributes, that are system defined or assigned by a standard value are not listed in the XML structure.
  The member attributes <ReadOnly> and <Informative> are only written to the XML export file if their value is TRUE.

- StartValue
  The element <StartValue> is only written, if the default value of the tag or constant is set by the user.

```
...
<Member Name="Static_1" Datatype="Int">
 ...
 <StartValue>1</StartValue>
</Member>
...
```

- Comment
  The element <Comment> is written, if it is set by the user. Comments of a tag or constant are exported as multilingual text:

```
...
 <Member Name="Static_3" Datatype="Struct">
   <AttributeList>
     <BooleanAttribute Name="ExternalAccessible" SystemDefined="true">false</BooleanAttribute>
     <BooleanAttribute Name="ExternalVisible" SystemDefined="true">false</BooleanAttribute>
     <BooleanAttribute Name="ExternalWritable" SystemDefined="true">false</BooleanAttribute>
   </AttributeList>
   <Comment>
     <MultiLanguageText Lang="de-DE">An individual comment</MultiLanguageText>
   </Comment>
 </Member>
...
```

Attributes

- Main attributes
  The main attributes are written in the <Member> element in the XML structure.

```
...
<Member Name="Static_1" Datatype="&amp;quot;User_data_type_1&amp;quot;" Remanence="Retain">
  ...
</Member>
...
```

The following table shows the main attributes of a tag or constant at the block interface section.

Name	Datatype	Default	Import condition	Comment
Name	STRING	-	Required	
Datatype	ENUM	-	Required	
Version	STRING	-	Optional	
Remanence	ENUM	NonRetain	-	only written if not default
Accessibility	ENUM	Public	-	pre-defined by the system cannot be changed by the user
Informative	BOOL	FALSE	-	

Members with the flag "Informative" are ignored during import. If the attribute is deleted or set to FALSE, an exception is thrown.

Note

Remanence settings "Set in IDB"

If the remanence value of a tag or constant is "Set in IDB", the remanence set in the IDB has to be the same for all other tags and constants with the remanence value "SetInIDB".

The first imported member with "Set in IDB" attribute defines the expected remanence in the IDB for the following tags and constants with the remanence value "SetInIDB".

- System defined member attributes
  Systemdefined member attributes are listed in the element <AttributeList>.
  Systemdefined member attributes flagged with the <Informative> and are ignored during import.

```
...
<Member Name="Static_3" Datatype="Struct">
  <AttributeList>
    <BooleanAttribute Name="ExternalAccessible" SystemDefined="true">false</BooleanAttribute>
    <BooleanAttribute Name="ExternalVisible" SystemDefined="true">false</BooleanAttribute>
    <BooleanAttribute Name="ExternalWritable" SystemDefined="true">false</BooleanAttribute>
  </AttributeList>
  <Comment>
    <MultiLanguageText Lang="de-DE">An individual comment</MultiLanguageText>
  </Comment>
</Member>
...
```

Name	Type	Default	SimaticML ReadOnly (informative)	Comment
At	string	""	FALSE	Member shares offset with another member in this structure
SetPoint	bool	FALSE	FALSE	Member can be synchronized with workmemory
UserReadOnly	bool	FALSE	TRUE	User cannot change any member attribute (incl. name)
UserDeletable	bool	TRUE	TRUE	Editor does not allow to delete the member
HmiAccessible	bool	TRUE	FALSE	No HMI access, no structure item
HmiVisible	bool	TRUE	FALSE	Filter to reduce the number of members shown in the first place
Offset	int	-	TRUE	DB, FB, FC (Temp). For classic PLCs and for Plus PLCs where the remanence is set to classic.
PaddedSize	int	-	TRUE	DB, FB, FC (Temp). For classic PLCs and for Plus PLCs where the remanence is set to classic. Only for arrays.
HiddenAssignment	bool	FALSE	FALSE	Hide assignement at call if matches with PredefinedAssignment
PredefinedAssingment	string	""	FALSE	Input for the paramter used when call is placed
ReadOnlyAssignment	bool	FALSE	FALSE	The user cannot change the predefined assignement at the call
UserVisible	bool	TRUE	TRUE	This member is not shown on the UI

Name	Type	Default	SimaticML ReadOnly (informative)	Comment
HmiReadOnly	bool	TRUE	TRUE	This member is read only for HMI
CodeReadOnly	bool	FALSE	TRUE	-

- User defined attributes
  User defined attributes are flagged with <ReadOnly>. Members with this flag are ignored during import. If the flag is deleted or set to FALSE, an exception is thrown.
  Unedited user defined attributes are excluded from the export.

Name	Type	Default	SimaticML ReadOnly (informative)	Comment
CFC	IBlockAttribute	---	FALSE	this is a Payload

Datatype "STRUCT"

The components of the datatype "STRUCT" are represented in the XML structure of an import/export file as nested members:

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Struct">
      <!-- Basic struct -->
      <Member Name="Static_1" Datatype="Int">
        <!-- First Member of struct -->
        <StartValue>1</StartValue>
      </Member>
      <Member Name="Static_2" Datatype="Int">
        <!-- Second Member of struct -->
      </Member>
      <Member Name="Static_3" Datatype="Struct">
        <!-- A subsequent struct -->
        <Member Name="Static_1" Datatype="Int">
          <!-- First Member of the subsequent struct -->
          <StartValue>3</StartValue>
        </Member>
        <Member Name="Static_2" Datatype="Int">
          <!-- Second Member of the subsequent struct -->
        </Member>
      </Member>
    </Member>
  </Section>
</Sections>
```

Datatype "ARRAY" basic type

The components of the basic datatype "ARRAY" are represented in the XML structure of an import/export file as subelements with the attribute "`Path`" :

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Array[0..2] of Int" Remanence="Retain">
      <!-- Basic Array -->
      <Subelement Path="0">
        <!-- First Array Component-->
        <StartValue>1</StartValue>
      </Subelement>
      <Subelement Path="1">
        <!-- Second Array Component-->
        <StartValue>2</StartValue>
      </Subelement>
    </Member>
  </Section>
</Sections>
```

Datatype "ARRAY" of UDT

The components of the datatype "ARRAY" of an UDT are represented in the XML structure of an import/export file as new `<sections>` element in a `<member>` element. The members in the new section for UDT are within an ARRAY are assigned as subelements with "`Path`" attribute:

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="&amp;quot;User_data_type_1&amp;quot;" Remanence="Retain">
      <Sections> <!-- Sections including the UDT "User_data_type_1" -->
        <Section Name="None">
          <Member Name="Element_2" Datatype="Int">
            <StartValue>47</StartValue>
          </Member>
        </Section>
      </Sections>
    </Member>
    <Member Name="Static_2" Datatype="Array[0..1] of &amp;quot;User_data_type_1&amp;quot;">
      <Sections> <!-- Sections including the UDT "User_data_type_1" -->
        <Section Name="None">
          <Member Name="Element_2" Datatype="Int">
            <Subelement Path="0">  <!-- Component of the array -->
              <StartValue>123</StartValue>
            </Subelement>
          </Member>
        </Section>
      </Sections>
    </Member>
  </Section>
</Sections>
```

Datatype "ARRAY" in "ARRAY"

The components of the datatype "ARRAY" in another ARRAY are represented in the XML structure of an import/export file as subelements with the attribute "`Path`".

The members within another ARRAY are assigned as subelements with"`Path`" attribute, if the component is edited by the user:

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Array[0..2] of Struct">
      <Member Name="Static_1" Datatype="Int" />
      <Member Name="Static_2" Datatype="Array[0..1, 0..3, -9..-2] of Struct">
        <Member Name="Static_1" Datatype="Int">
          <Subelement Path="0,0,3,-5">
            <StartValue>1</StartValue>
          </Subelement>
        </Member>
        <Subelement Path="0,0,2,-6">
          <Comment>
            <MultiLanguageText Lang="de-DE">A individual comment</MultiLanguageText>
          </Comment>
        </Subelement>
      </Member>
    </Member>
  </Section>
</Sections>
```

PLC data types (UDT)

The XML structure of a PLC data type depends on the TIA Portal Openness export settings.

- `ExportOptions.None`
  Members of PLC data type are only written if the default value of at least one of the components is set by the user. For these members, only the two additional attributes "`Name`" and "`Datatype`" are written, to identify the member to which the `<StartValue>` belongs. Other members and attributes are not written.
- `ExportOptions.WithDefaults`
  The following attributes are always written:
  - `Name`
  - `Datatype`
  - `ExternalAccessible`
  - `ExternalVisible`
  - `ExternalWritable`
  - `SetPoint`
  - `StartValue`
    Only written to the XML if it the default value in this type is set by the user. If it only has been set in the PLC data type, it is not written.
- `ExportOptions.ReadOnly`
  For PLC data types this setting will not lead to meaningful result. In combination with other settings it will have no influence on the result.

Overlaid tags

If a tag is overlaid with a new datatype, the members are represented in the XML structure of the new data type. The following XML structure shows a datatype WORD overlaid by an ARRAY of BYTE.

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Input" />
  <Section Name="Output" />
  <Section Name="InOut" />
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Word">
      <!-- Basic Member -->
      <StartValue>16#17</StartValue>
    </Member>
    <Member Name="Static_2" Datatype="Array[0..1] of Byte">
      <AttributeList>
        <StringAttribute Name="At" SystemDefined="true">Static_1</StringAttribute>
      </AttributeList>
        <!-- The AT member -->
      <Subelement Path="0">
        <!-- First overlay byte -->
      </Subelement>
      <Subelement Path="1">
        <!-- Second overlay byte -->
      </Subelement>
    </Member>
  </Section>
  <Section Name="Temp" />
  <Section Name="Constant" />
</Sections>
```

Block Interface

All attributes with `ReadOnly`="TRUE" and `Informative`="FALSE" are excluded. The XML structure of a block interface depends on the TIA Portal Openness export settings.

- `ExportOptions.None`
  This setting exports only the modified data or the data that differs from the default.
  In case their attribute definition does not specify a default value, the attribute is always written.
  The export file also contains all values that are obligatory for the subsequent data import.
- `ExportOptions.WithDefaults`
  The following attributes are always written
  - `Name`
  - `Datatype`
  - `HmiAccessible` exported as `ExternalAccessible`
  - `HmiVisible` exported as `ExternalVisible`
  - `ExternalWritable`
  - `SetPoint` (if applicable)
  - `Offset` (if applicable)
  - `PaddedSize` (if applicable)

  All other attributes are only written if their values differ from the default.
  The `<StartValue>` element is only written to the XML if it has been explicitly set.
- `ExportOptions.ReadOnly`
  For block interfaces this setting will not lead to meaningful result. In combination with other settings it will have no influence on the result.

6.4.2.2 Changes of the object model and XML file format

Introduction

To import a custom created or an edited XML file successfully to the TIA Portal via TIA Portal Openness, the file must correspond to defined schemas.

The XML files always consist of two major parts:

- Interface
- Compile unit

The schemas they have to correspond to are explained in the following.

Interface

An interface can contain multiple sections (e. g. Input, InOut, Static): You can find all of these sections in the schema in the following directory:

- C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.InterfaceSections_v3.xsd
- C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.Interface.Snapshot .xsd

Compile unit

There are seperate schemas for the compile units of GRAPH, LAD/FBD, STL and SCL blocks. You can find these schemas in the following directories:

- GRAPH: C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.Graph_v4.xsd
- LAD/FBD: C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.LADFBD_v3.xsd
- STL: C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.STL_v3.xsd
- SCL: C:\Pogram Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\ SW.PlcBlocks.SCL_v2.xsd

Subschemas

There are the following additional schema definitions used by all compile units:

- Access
- Common

Access

The Access node describes for example:

- local/global members and constant usages
- FB, FC, Instruction calls
- DBs for calls

You can find the access schema in the following directory:

C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.Access_v3.xsd

Common

Common contains the commonly used attributes and elements, for example different types of comments, texts and tokens.

You can find the common schema in the following directory:

C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*.\Schemas\SW.Common_v2.xsd

Note

V* refers to the installed version of TIA Portal.

6.4.2.3 Exporting DBs with snapshots

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to export the DB's with snapshot value in XML format to compare the values with different snapshot times. With the compare result, you can manually adapt

single start values within the TIA Portal user interface and store for a potential recovery.

The schema “SW.Interface.Snapshot.xsd” can process the exported XML-file. The Snapshot Service "InterfaceSnapshot" is provided in the namespace "Siemens.Engineering.SW.Blocks".

The file handling supports exporting DBs with snapshots:

- Export directory does not exist
- Creating of the export directory
- Export directory is readonly
- If export file already exists

The Snapshot Service is supported for Global DBs, Instance DBs and Array DB

Program code

Modify the following program code to export Snapshot Values by using the Snapshot Service:

```
PlcBlock dataBlock = ...;
InterfaceSnapshot interfaceSnapshot = dataBlock.GetService<InterfaceSnapshot>();
interfaceSnapshot.Export(new FileInfo("C:\\temp\\MyInterfaceSnapshot.xml"),
ExportOptions.WithReadOnly);
```

Note

Export of snapshot values using the Snapshot Service is independent of the standard interface Openness export and will not influence the already existing export of the interface members. It will not be possible to import the exported XML

The snapshot values will be exported as following:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
<Engineering version="V15 SP1" />
<DocumentInfo>
...
</DocumentInfo>
<SW.Blocks.InterfaceSnapshot ID="0">
<AttributeList>
<Name>GlobalDB</Name>
<Snapshot ReadOnly="true">
<SnapshotValues>
<Value Path="Static_1" Type="Bool">TRUE</Value>
<Value Path="Static_2[0]" Type="Int">1</Value>
<Value Path="Static_2[1]" Type="Int">2</Value>
<Value Path="Static_2[2]" Type="Int">3</Value>
<Value Path="Static_3" Type="DTL">DTL#1973-01-01-00:00:00</Value>
<Value Path="Static_4.Element_1" Type="Int">7</Value>
<Value Path="Static_4.Element_2[0]" Type="Bool">FALSE</Value>
<Value Path="Static_4.Element_2[1]" Type="Bool">TRUE</Value>
<Value Path="Static_4.Element_2[2]" Type="Bool">TRUE</Value>
<Value Path="Static_4.Element_3.Element_1" Type="Int">5</Value>
<Value Path="Static_4.Element_3.Element_2.Element_1" Type="Bool">TRUE</Value>
<Value Path="Static_4.Element_3.Element_2.Element_2[0]" Type="Int">100</Value>
<Value Path="Static_4.Element_3.Element_2.Element_2[1]" Type="Int">200</Value>
</SnapshotValues></Snapshot>
<SnapshotDate ReadOnly="true">2017-12-06T08:04:11.4590585Z</SnapshotDate>
<StructureModified ReadOnly="true">2017-12-06T08:22:13.3292585Z</StructureModified>
</AttributeList>
</SW.Blocks.InterfaceSnapshot>
</Document>
```

If a DB does not contain any snapshot values, the content of the exported file will look like as following:

```
<SnapshotValues xlmns="http://www.siemens.com/automation/Openness/SW/Interface/Snapshot/
v1"></SnapshotValues>
```

See also

Opening a project (Page 128)

6.4.2.4 Exporting blocks with know-how protection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Introduction

The resulting XML file is similar to the export file of a block without know-how protection. With PlcBlockProtectionProvider, you can unlock a know-how protected block by providing the password using the Openness API and then export a KHP block with in complete with its code. A block can be imported and then be protected with a password using the API again.

If the block is exported without unlocking, only the public block interface will be exported.

- TIA Portal block -> Unlock -> Export file without protection
- Import file without protection -> Lock -> TIA Portal block

The attribute list of the block indicates that the relevant block is know-how protected.

Program code

Modify the following program code to export the visible data of a block with know-how protection to an XML file:

```
private static void ExportBlock(PlcSoftware plcSoftware)
{
PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
plcBlock.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", plcBlock.Name)),
ExportOptions.WithDefaults);
}
```

6.4.2.5 Export/Import of SCL blocks

SCL statements with export XML tags

The export operation of SCL blocks exports its equivalent XML tags based on the type of SCL statements. This operation supports the SCL networks of SCL statements in LAD/FBD blocks of SCL statements. The SCL statements are classified as text elements, operands, expressions, control, etc. The schema SW.PlcBlocks.SCL_v2.xsd is available to support you to process the SCL block with export XML tags. The SCL block statements with their corresponding exported XML tags and attributes are given below.

New line

New lines in SCL blocks are represented as NewLine XML tag.

- Contains unsigned Num attribute with default value 1.
- Num attribute does not have value 0.
- Supported only for SCL.

SCL block	XML tag
	`<NewLine Num="2" />`

Blank

Blank spaces in SCL blocks are respresented as Blank XML tag.

- Contains unsigned Num attribute with default value 1.
- Num attribute does not have value 0.
- Supported only for SCL.
- Does not support Integer attribute available in other languages of STEP 7.

SCL block	XML tag
	`<Blank Num="2"/>`

Identation of SCL block statements

In TIA Portal settings, you can modify the identation of SCL code by accessing Options/Settings/ General/Script/text editiors. The following table defines the type of identation based on the ident mode.

Ident mode	Result
None	Import operation adds the spaces as available in the source files.
Paragraph or Smart	Import operation adds the specified ident spaces in the imported file.

Based on the chosen identation, the imported SCL block XML file are idented.

Comment

Single-line and multi-line comments in SCL blocks are represented as LineComment XML tag.

- Only LineComment tag (for single language comment) is used in SCL.
- Comment tag (for mulitple language comment) is not used in SCL.
- Contains Inserted attribute with default value false
- Inserted="false" indicates "//" single comment in SCL block.
- Inserted="true" indicates "(**)" multi-line comment in SCL block.

- NoClosingBracket="true" indicates comment without closing braces in SCL block. This attribute is optional and has default value as false.
- XML does not indicate comment hierarchy in SCL block.

SCL block	XML tag

`// one line comment`	`<LineComment>`
	`  <Text>one line comment</Text>`
	`</LineComment>`

`(* one line comment`
`second line *)`
	`<LineComment Inserted="true">`
	`  <Text>one linecomment`
	`secondline</Text>`
	`</LineComment>`

`(* first comment (* second comment`
`*) end first comment *)`
	`<LineComment Inserted=”true”>`
	`  <Text> first comment (*`
	`second comment *) end first comment</`
	`Text>`
	`</LineComment >`
	The nested comment is part of outer comment text.

`(* comment without closing bracket`	`<LineComment Inserted="true"`
	`NoClosingBracket="true">`
	`  <Text> comment without`
	`closing bracket</Text>`
	`</LineComment >`

Region

Regions in SCL blocks are represented as Token XML tag.

- Text XML tag represents the region_name.
- The Text attribute of the Token XML tag is case in-sensitive.
- The import operation is case in-sensitive, and the editor displays the keywords as configured in the TIA Portal settings.
- If the end_region keyword ends with ";" (semi-colon) in SCL block, the symbol ";" is placed in Text XML tag.

SCL block	XML tag
region myregion ... end_region here is the end of myregion	<Token Text="REGION" /> <Blank /> <Text>myregion</Text> <NewLine /> ... <Token Text="END_REGION" /> <Blank /> <Text>here is the end of myregion</Text> <NewLine />
region // here are no blanks ... end_region	<Token Text="REGION" /> <NewLine /> <LineComment .../> <Token Text="END_REGION" /> <NewLine />
region ... end_region;	<Token Text="REGION" /> <NewLine /> ... <Token Text="END_REGION" /> <Text>;</Text> <NewLine />

Pragma

Pragma in SCL blocks are represented as Token XML tag. The parameters are represented in Access XML tag with Scope attribute as LiteralConstant.

SCL block	XML tag
{PRAGMA_BEGIN 'Param1', 'Param2' (*parm 2*)} // something else {PRAGMA_END}	<Token Text="{" /> <Token Text="PRAGMA_BEGIN" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>'Param1'</ ConstantValue> </Constant> </Access> <Token Text="," /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>'Param2'</ ConstantValue> </Constant> </Access> <Blank /> <LineComment Inserted="True"> <Text>param 2</Text> </LineComment> <Token Text="," /> <Blank /> <Token Text="}" /> <NewLine /> <LineComment> <Text> something else</Text> </LineComment> <NewLine /> <Token Text="{" /> <Token Text="PRAGMA_END" /> <Token Text="}" />

Constants: Literal constants

The constants in SCL blocks are represented by Access XML tag.

- The Scope attribute can have values like LiteralConstant, TypedConstant, LocalConstant, and GlobalConstant.
- The name of constants preceeded by "#" are ignored in XML.
- The "#" is added during the import operation of XML.
- The value of global constants represented by quotes are ignored in XML.
- The quotes are added during the import operation of XML.

Type of constant	SCL block	XML tag

Literal constant: Integer	`#Out := 10;`	`<Access Scope="LiteralConstant">`
		`  <Constant>`
		`  <ConstantValue>10</ConstantValue>`
		`  <ConstantTypeInformative="true">LINT</ConstantType>`
		`  </Constant>`
		`</Access>`

Literal constant: String	`#myString := 'Hello world';`	`<Access Scope="LiteralConstant">`
		`  <Constant>`
		`  <ConstantValue>Hello world</ConstantValue>`
		
		`<ConstantTypeInformative="true">STRING</ConstantType>`
		`  </Constant>`
		`</Access>`

Literal constant: Typed	`#Out := int#10;`	`<Access Scope="TypedConstant">`
		`  <Constant>`
		`  <ConstantValue>int#10</ConstantValue>`
		`  </Constant>`
		`</Access>`

		Format of XML exported in ExportOptions.ReadOnly setting.
		
		`<Access Scope="TypedConstant">`
		`  <Constant>`
		`  <ConstantValue>int#10</ConstantValue>`
		`  <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute>`
		`  <StringAttribute Name="FormatFlags" Informative="true">TypeQualifier</StringAttribute>`
		`  </Constant>`
		`</Access>`

Local constant	`#Out := #mylocal;`	`<Access Scope="LocalConstant">`
		`  <Constant Name="mylocal" />`
		`</Access>`

		Format of XML exported in ExportOptions.ReadOnly setting.
		
		`<Access Scope="LocalConstant">`
		`  <Constant Name="mylocal">`
		`  <ConstantType Informative="true">Int</ConstantType>`
		`  <ConstantValue Informative="true">10</ConstantValue>`
		`  <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute>`
		`  </Constant>`
		`</Access>`

Type of constant	SCL block	XML tag

Global constant	`#Out := "myglobal";`	`<Access Scope="GlobalConstant">`
		`  <Constant Name="myglobal" />`
		`</Access>`

		Format of XML exported in ExportOptions.ReadOnly setting.
		
		`<Access Scope="GlobalConstant">`
		`  <Constant Name="myglobal">`
		`  <ConstantType Informative="true">Int</ConstantType>`
		`  <ConstantValue Informative="true">10</ConstantValue>`
		`  <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute>`
		`  </Constant>`
		`</Access>`

The address constants are not supported in SCL blocks, and it is ignored in this table.

Variables

The local and global variables in SCL blocks are represented by Access XML tag.

- The Scope attribute has values of LocalVariable and GlobalVariable
- The XML tags for assigning the value 10 is ignored here.

Type of variable	SCL block	XML tag

Local variable	`#Out := 10;`	`<Access Scope="LocalVariable">`
		`  <Symbol>`
		`  <Component Name="Out" />`
		`  </Symbol>`
		`</Access>`

Global variable	`"Tag_3":= 10;`	`<Access Scope="GlobalVariable">`
		`  <Symbol>`
		`  <Component Name="Tag_3" />`
		`  </Symbol>`
		`</Access>`

		Format of XML exported in ExportOptions.ReadOnly setting.
		
		`<Access Scope="GlobalVariable">`
		`  <Symbol>`
		`  <Component Name="Tag_3" />`
		`  <Address Area="Memory" Type="Int" BitOffset="96" Informative="true" />`
		`  </Symbol>`
		`</Access>`

Expressions

The simple expressions in SCL blocks are represented by Access XML tag. The Scope attribute has value of LocalVariable for the expressions

SCL block	XML tag

`#a := #b + #c;`	`<Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="a" />`
	`  </Symbol>`
	`</Access>`
	`<Blank />`
	`<Token text=":=" />`
	`<Blank />`
	`<Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="b" />`
	`  </Symbol>`
	`</Access>`
	`<Blank />`
	`<Token text="+" />`
	`<Blank />`
	`<Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="c" />`
	`  </Symbol>`
	`</Access>`
	`<Token text=";" />`

Control structures in SCL blocks

The control statements like IF, CASE, FOR, WHILE, REPEAT, GOTO, EXIT, CONTINE, and RETURN are represented by Token XML tag.

- The conditional symbols used in SCL block such as >, <, & are represented as escape sequences (< > &amp) in XML.
- These combination of XML tags are applicable only for SCL blocks. An exception is thrown for other languages.

Name of the block	SCL block	XML tag
IF	IF #a<#c THEN ; END_IF;	<Token Text="IF" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token Text="<" /> <Access Scope="LocalVariable"> <Symbol> <Component Name="c" /> </Symbol> </Access> <Blank /> <Token Text="THEN" /> <NewLine /> <Blank Num="4" /> <Token Text=";" /> <NewLine /> <Token Text="END_IF" /> <Token Text=";" />

Name of the block	SCL block	XML tag

CASE	CASE #a OF
	  1 (*test*): // Statement section case 1
	  ;
	  2..4: // Statement section case 2 to 4
	  ;
	  ELSE // Statement section ELSE
	  ;
	END_CASE;
		<Tok en Text="CASE" /><Blank />
		<Access Scope="LocalVariable">
		  <Symbol>
		  <Component Name="a" />
		  </Symbol>
		</Access>
		<Blank />
		<Token Text="OF" />
		<NewLine />
		
		<Blank Num="2"/>
		<Access Scope="LiteralConstant">
		  <Constant>
		  <ConstantValue>1</ConstantValue>
		  <ConstantType Informative="true">LINT</ConstantType>
		  </Constant>
		</Access>
		<Blank />
		<LineComment Inserted=”true”>
		  <Text>test</Text>
		</LineComment >
		<Token Text=":" />
		<Blank />
		<LineComment>
		  <Text> Statement section case 1</Text>
		</LineComment >
		<NewLine />
		
		<Blank Num="4"/>
		<Token Text=";" />
		<NewLine />
		
		<Blank Num="2"/>
		<Access Scope="LiteralConstant">
		  <Constant>
		  <ConstantValue>2</ConstantValue>
		  <ConstantType Informative="true">LINT</ConstantType>
		  </Constant>
		</Access>
		<Token Text=".." />
		<Blank Num="2"/>
		<Access Scope="LiteralConstant">
		  <Constant>
		  <ConstantValue>4</ConstantValue>
		  <ConstantType Informative="true">LINT</ConstantType>
		  </Constant>
		</Access>
		<Blank />
		<LineComment>
		  <Text> Statement section case 2 to 4</Text>
		</LineComment >
		<NewLine />

Name of the block	SCL block	XML tag
		<Blank Num="4"/> <Token Text=";" /> <NewLine /> <Blank Num="2"/> <Token Text="ELSE" /> <NewLine /> <Blank Num="4"/> <Token Text=";" /> <NewLine /> <Token Text="END_CASE" /> <Token Text=";" />

FOR	FOR #i := #a TO #b DO
	  // Statement section FOR
	  ;
	END_FOR;
		<Token Text="FOR" />
		<Blank />
		<Access Scope="LocalVariable">
		  <Symbol>
		  <Component Name="i" />
		  </Symbol>
		</Access>
		<Blank />
		<Token Text=":=" />
		<Blank />
		<Access Scope="LocalVariable">
		  <Symbol>
		  <Component Name="a" />
		  </Symbol>
		</Access>
		<Blank />
		<Token Text="TO" />
		<Blank />
		<Access Scope="LocalVariable">
		  <Symbol>
		  <Component Name="b" />
		  </Symbol>
		</Access>
		<Blank />
		<Token Text="DO" />
		<NewLine />
		
		<Blank Num="2" />
		<LineComment>
		  <Text> Statement section FOR</Text>
		</LineComment >
		<NewLine />
		
		<Blank Num="2" />
		<Token Text=";" />
		<NewLine />
		
		<Token Text="END_FOR" />
		<Token Text=";" />

Name of the block	SCL block	XML tag

WHILE	`WHILE #a<#b DO`
	`  // Statement section WHILE`
	`  ;`
	`END_WHILE;`
		`<Token Text="WHILE" />`
		`<Blank />`
		`<Access Scope="LocalVariable">`
		`  <Symbol>`
		`  <Component Name="a" />`
		`  </Symbol>`
		`</Access>`
		`<Token Text="<" />`
		`<Access Scope="LocalVariable">`
		`  <Symbol>`
		`  <Component Name="b" />`
		`  </Symbol>`
		`</Access>`
		`<Blank />`
		`<Token Text="DO" />`
		`<NewLine />`
		
		`<Blank Num="2" />`
		`<LineComment>`
		`  <Text> Statement section WHILE</Text>`
		`</LineComment >`
		`<NewLine />`
		
		`<Blank Num="2" />`
		`<Token Text=";" />`
		`<NewLine />`
		
		`<Token Text="END_WHILE" />`
		`<Token Text=";" />`

Name of the block	SCL block	XML tag

REPEAT	REPEAT
	  // Statement section REPEAT
	  ;
	UNTIL #a<#b END_REPEAT;
		<Token Text="REPEAT" />
		<NewLine />
		
		<Blank Num="2" />
		<LineComment>
		  <Text> Statement section REPEAT</Text>
		</LineComment >
		<NewLine />
		
		<Blank Num="2" />
		<Token Text=";" />
		<NewLine />
		
		<Token Text="UNTIL" />
		<Blank />
		<Access Scope="LocalVariable">
		  <Symbol>
		  <Component Name="a" />
		  </Symbol>
		</Access>
		<Token Text="<" />
		<Access Scope="LocalVariable">
		  <Symbol>
		  <Component Name="b" />
		  </Symbol>
		</Access>
		<Blank />
		<Token Text="END_REPEAT" />
		<Token Text=";" />

Name of the block	SCL block	XML tag
GOTO	here // well : // this is goto statement	XML example for GOTO label definition <Blank Num="3"/> <Access Scope="Label"> <Label Name="here"> <NewLine /> <Blank Num="3"/> <LineComment> <Text> well</Text> </LineComment> <NewLine /> <Token Text=":" /> <Blank /> </Label> </Access> <LineComment> <Text> this is goto statement</Text> </LineComment>
	GOTO (*comment*) here;	XML example for GOTO label usage <Token Text="GOTO" /> <Blank /> <LineComment inserted=”true”> <Text>comment</Text> </LineComment> <Blank /> <Access Scope="Label"> <Label Name="here" /> </Access> <Token Text=";" />

Referencing attributes

The SCL block referencing attributes are represented by AccessModifier attribute of Component tag.

- For simple referencing, AccessModifer has value as Reference.
- For array referencing, AccessModifier has value as ReferenceToArray.

SCL block	XML tag

`RefToUDT^(*RefToUDT*).element`	`<Symbol>`
	`  <Component Name="RefToUDT"`
	`AccessModifier="Reference" />`
	`  <Token Text="^" />`
	`  <LineComment Inserted="True">`
	`  <Text>RefToUDT</Text>`
	`  </LineComment>`
	`  <Token Text="." />`
	`  <Component Name="element" />`
	`</Symbol>`

RefToArrayOfUDT^(*RefToArrayOfUDT*)[#i].element	`<Symbol>`
	`  <Component Name="RefToArrayOfUDT"`
	`AccessModifier="ReferenceToArray" />`
	`  <Token Text="^" />`
	`  <LineComment Inserted="True">`
	`  <Text>RefToArrayOfUDT</Text>`
	`  </LineComment>`
	`  <Token Text="[" />`
	`  <Access Scope=LocalVariable>`
	`  <Symbol>`
	`  <Component Name="i" />`
	`  </Symbol>`
	`  </Access>`
	`  <Token Text="]" />`
	`</Component>`
	`  <Token Text="." />`
	`  <Component Name="element" />`
	`</Symbol>`

6.4.2.6 Export/Import of structured types of SCL blocks

SCL structured types with export XML tags

In the SCL structured types, you can add blanks, new lines, and comments in the SCL statements. The SCL strctured statements with its corresponding exported XML tags and attributes are given below.

Global access

In SCL statements, the global access variables and constants are represented in quotes. The comments written between the variables and address parts are represented by LineComment XML tag.

SCL block	XML tag

`"Data_block_1".(*comment 1*)Static_1(*comment 2*).Static_2`	`<Access Scope="GlobalVariable">`
	`  <Symbol>`
	`  <Component Name="Data_block_1" />`
	`  <Token Text="." />`
	`  <LineComment Inserted="True">`
	`  <Text>comment 1</Text>`
	`  </LineComment>`
	`  <Component Name="Static_1" />`
	`  <LineComment Inserted="True">`
	`  <Text>comment 2</Text>`
	`  </LineComment>`
	`  <Token Text="." />`
	`  <Component Name="Static_2" />`
	`  </Symbol>`
	`</Access>`

`"Data_block_1".Static_1 := 10`	Format of XML exported in ExportOptions.None setting.
	
	`<Access Scope="GlobalVariable">`
	`  <Symbol>`
	`  <Component Name="Data_block_1" />`
	`  <Token Text="." />`
	`  <Component Name="Static_1" />`
	`  </Symbol>`
	`</Access>`

	Format of XML exported in ExportOptions.ReadOnly setting.
	
	`<Access Scope="GlobalVariable">`
	`  <Symbol>`
	`  <Component Name="Data_block_1" />`
	`  <Token Text="." />`
	`  <Component Name="Static_1" />`
	`  <Address Area="DB" Type="Word" BlockNumber="1" BitOffset="0" Informative="true" />`
	`  </Symbol>`
	`</Access>`

Usage of Quotes and

The quotes used in the first level describes the type of variable, and used to escape special characters in SCL statements. When quotes are used in first level, it defines the variable as global variable. If the quotes are used after #, they represent the escape sequence of special characters like #, and spaces.

- To represent the differential usage, XML file uses BooleanAttributes tag with Name attribute. The Name contain values such as HasQuotes and HasHash.
- To define structure in scope attribute, # is defined.
- These values are applicable only for SCL.
- The default values for these tags were FALSE, but the values never gets exported in ExportOptions.WithDefaults settings too.

SCL block	XML tag

`"a".#b."c".#"d"`	`<Access Scope="GlobalVariable">`
	`  <Symbol>`
	`  <Component Name="a" />`
	`  <Token Text="." />`
	`  <Component Name="b">`
	`  <BooleanAttribute Name=”HasHash”>TRUE</BooleanAttribute>`
	`  </Component>`
	`  <Token Text="." />`
	`  <Component Name="c">`
	`  <BooleanAttribute Name=”HasQuotes”>TRUE</BooleanAttribute>`
	`  </Component>`
	`  <Token Text="." />`
	`  <Component Name="d">`
	`  <BooleanAttribute Name=”HasQuotes”>TRUE</BooleanAttribute>`
	`  <BooleanAttribute Name=”HasHash”>TRUE</BooleanAttribute>`
	`  </Component>`
	`  </Symbol>`
	`<Access />`

Array

SCL allows to add comment within the array indexes around "[" and "]". To mark the existence of array, XML file uses AccessModifier attribute in Component tag.

- If Accessmodifier contains the value Array, then a child tag Access is mandatoy to indicate the index variable of the array.
- The default value for AccessModifier is None.

SCL block	XML tag

`#a.b[#i+#j,#k+#l].c`	`<Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="a" />`
	`  <Token Text="." />`
	`  <Component Name="b"`
	`AccessModifier="Array" />`
	`  <Token Text="[" />`
	`  <Access Scope=LocalVariable>`
	`  <Symbol>`
	`  <Component Name="i" />`
	`  </Symbol>`
	`  </Access>`
	`  <Token Text="+" />`
	`  <Access Scope=LocalVariable>`
	`  <Symbol>`
	`  <Component Name="j" />`
	`  </Symbol>`
	`  </Access>`
	`  <Token Text="," />`
	`  <Access Scope=LocalVariable>`
	`  <Symbol>`
	`  <Component Name="k" />`
	`  </Symbol>`
	`  </Access>`
	`  <Token Text="+" />`
	`  <Access Scope=LocalVariable>`
	`  <Symbol>`
	`  <Component Name="l" />`
	`  </Symbol>`
	`  </Access>`
	`  <Token Text="]" />`
	`  </Component>`
	`  <Token Text="." />`
	`  <Component Name="c" />`
	`  </Symbol>`
	`</Access>`

Absolute Access

SCL allows different types of access such as absolute, absolute offset, mixed (database and member variable), slice, peripheral, and direct type. The absolute access specifiers are represented by Address tag in XML.

- The % character of the DB is not written in XML. It is created automatically during the import.
- Blanks are allowed between the address parts

SCL Block	XML Tag
%DB20 . DBW10	<Access Scope="Address"> <Symbol> <Address Area="DB" BlockNumber="20" /> <Blank /> <Token Text="." /> <Blank /> <Address Area="DB" BitOffset="80" Type="Word"/> </Symbol> </Access>
%DB20.DBX10.3 := true;	The following XML is valid for all langages except SCL. <Access Scope="Address"> <Address Area="DB" BlockNumber="20" BitOffset="83" Type="Bool" /> </Access> The following XML is valid for SCL. <Access Scope="Address"> <Symbol> <Address Area="DB" BlockNumber="20" /> <Token Text="." /> <Address Area="DB" BitOffset="83" Type="Bool"/> </Symbol> </Access>

Absolute offset

In STL, AbsoluteOffset tag represents the absolute offset access. In SCL, Address tag is used for absolute access.

SCL Block	XML Tag
#Input_DB_ANY.%DBX2.3 := TRUE;	<Access Scope="LocalVariable"> <Symbol> <Component Name="Input_DB_ANY" /> <Token Name="." /> <Address BitOffset="19" Type="Bool" /> </Symbol> </Access>

Slicing

In SCL, the SliceAccessModifier attribute is not supported and the slicing is represented by Token tag.

SCL Block	XML Tag
`"tag_1"(*1*).(*2*)member(*3*).(*4*) %x1`	`<Access Scope="GlobalVariable"> <Symbol> <Component Name="tag_1" /> <LineComment Inserted="True"> <Text>1</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>2</Text> </LineComment> <Component Name="member"/> <LineComment Inserted="True"> <Text>3</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>4</Text> </LineComment> <Token Text="%x1" /> </Symbol> </Access>`

Peripheral access

The peripheral access is represented by Token tag.

SCL Block	XML Tag
`"tag_1"(*1*).(*2*)member:P`	`<Access Scope="GlobalVariable"> <Symbol> <Component Name="tag_1" /> <LineComment Inserted="True"> <Text>1</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>2</Text> </LineComment> <Component Name="member"/> <Token Text=":P" /> </Symbol> </Access>`

Direct type access

The TypeOf and TypeOfDB instructions are handled either with system type or user defined type. The types are represented in Access tag with Scope attribute containing values of SystemType and UserType.

SCL Block	XML Tag
Example for system type `if TypeOf( #inVariant ) = TO_SpeedAxis then … end_if`	`<Token text="=" /> </Blank> <Access Scope="SystemType"> <DataType>TO_SpeedAxis</DataType> </Access>`
Example for user defined type `if TypeOf( #inVariant ) = "aUserDefinedType" then … end_if`	`<Token text="=" /> </Blank> <Access Scope="UserType"> <DataType>aUserDefinedType</DataType> </Access>`

6.4.2.7 Export/Import of SCL call blocks

SCL call blocks with export XML tags

SCL call paramaters are respresented by Parameter tag in XML. The informative attribute is used to represent the non-assigned FB parameters and return values such as timestamp, flag information, etc. The XML format follows the same arbitrary order followed in the SCL block.

An example for block call is given below.

SCL block	XML tag

`#Callee_Instance(Input_1 := 5);`	Format of XML exported in ExportOptions.None setting
	
	`<Access Scope="Call">`
	`  <CallInfo BlockType="FB">`
	`  <Instance Scope="LocalVariable">`
	`  <Component Name="Callee_Instance" />`
	`  </Instance>`
	`  <Token text="(" />`
	`  <Parameter Name="Input_1">`
	`  <Blank />`
	`  <Token text=":=" />`
	`  <Blank />`
	`  <Access Scope="LiteralConstant">`
	`  <Constant>`
	`  <ConstantType>Int</ConstantType>`
	`  <ConstantValue>5</ConstantValue>`
	`  </Constant>`
	`  </Access>`
	`  </Parameter>`
	`  <Token text=")" />`
	`  </CallInfo>`
	`</Access>`
	`  <Token text=";" />`

	Format of XML exported in ExportOptions.ReadOnly setting
	
	`<Access Scope="Call">`
	`  <CallInfo BlockType="FB">`
	`  <IntegerAttribute Name="BlockNumber" Informative="true">1</IntegerAttribute>`
	`  <DateAttribute Name="ParameterModifiedTS" Informative="true">2016-10-24T08:27:34</DateAttribute>`
	`  <Instance Scope="LocalVariable">`
	`  <Component Name="Callee_Instance" />`
	`  </Instance>`
	`  <Token text="(" />`
	`  <Parameter Name="Input_1">`
	`  <StringAttribute Name="InterfaceFlags" Informative="true">S7_Visible</StringAttribute>`
	`  <Blank />`
	`  <Token text=":=" />`

SCL block	XML tag
	<Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ConstantType> <ConstantValue>5</ConstantValue> </Constant> </Access> </Parameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" />

Unconnected parameters example

The FB has 4 paramters where a, b, c, and d. b and d are not connected.

SCL block	XML tag
"Block_4_DB"(a:=TRUE,c:=TRUE);	<Access Scope="Call"> **<CallInfo Name="Block_4" BlockType="FB">** <Instance Scope="GlobalVariable"> **<Component Name="Block_4_DB" />** </Instance> <Token text="(" /> **<Parameter Name="a">** **<Token text=":=" />** <Access Scope="LiteralConstant"> <Constant> <ConstantType>Bool</ConstantType> **<ConstantValue>TRUE</ConstantValue>** </Constant> </Access> </Parameter> <Token text="," /> <Parameter Name="b" Informative="true"/> **<Parameter Name="c" >** **<Token text=":=" />** <Access Scope="LiteralConstant"> <Constant> <ConstantType>Bool</ConstantType> **<ConstantValue>True</ConstantValue>** </Constant> </Access> </Parameter> <Parameter Name="d" Informative="true"/> <Token text=")" /> </CallInfo> </Access>

One parameter example

SCL block allows you to omit the parameter name. This parameter is represented as NamelessParameter tag. The NamelessParameter tag has no attributes and it is applicable only for SCL.

SCL block	XML tag

`"Block_4_DB"(TRUE);`	`<Access Scope="Call">`
	`  <CallInfo Name="Block_4"`
	`BlockType="FB">`
	`  <Instance`
	`Scope="GlobalVariable">`
	`  <Component`
	`Name="Block_4_DB" />`
	`  </Instance>`
	`  <Token text="(" />`
	`  <NamelessParameter>`
	`  <Access`
	`Scope="LiteralConstant">`
	`  <Constant>`
	`  <ConstantType>Bool</`
	`ConstantType>`
	`  <ConstantValue>TRUE</`
	`ConstantValue>`
	`  </Constant>`
	`  </Access>`
	`  </NamelessParameter>`
	`  <Token text=")" />`
	`  </CallInfo>`
	`</Access>`

Expression as actual parameter

SCL block	XML tag

`#Callee_Instance(Input_1 := #a+3);`	`<Access Scope="Call">`
	`  <CallInfo BlockType="FB">`
	`  <Instance Scope="LocalVariable">`
	`  <Component Name="Callee_Instance" />`
	`  </Instance>`
	`  <Token text="(" />`
	`  <Parameter Name="Input_1">`
	`  <Blank />`
	`  <Token text=":=" />`
	`  <Blank />`
	`  <Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="a" />`
	`  </Symbol >`
	`  </Access>`
	`  <Token text="+" />`
	`  <Access Scope="LiteralConstant">`
	`  <Constant>`
	`  <ConstantType>Int</ConstantType>`
	`  <ConstantValue>3</ConstantValue>`
	`  </Constant>`
	`  </Access>`
	`  </Parameter>`
	`  <Token text=")" />`
	`  </CallInfo>`
	`</Access>`
	`<Token text=";" />`

Expression as actual parameter without formal paramter

SCL block	XML tag

`#Callee_Instance(#a+3);`	`<Access Scope="Call">`
	`  <CallInfo BlockType="FB">`
	`  <Instance Scope="LocalVariable">`
	`  <Component Name="Callee_Instance" />`
	`  </Instance>`
	`  <Token text="(" />`
	`  <NamelessParameter>`
	`  <Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="a" />`
	`  </Symbol >`
	`  </Access>`
	`  <Token text="+" />`
	`  <Access Scope="LiteralConstant">`
	`  <Constant>`
	`  <ConstantType>Int</ConstantType>`
	`  <ConstantValue>3</ConstantValue>`
	`  </Constant>`
	`  </Access>`
	`  </NamelessParameter>`
	`  <Token text=")" />`
	`  </CallInfo>`
	`</Access>`
	`<Token text=";" />`

Function call

SCL block	XML tag

`#myInt := "MyFunction"(Param_1 := 1, Param_2 := 15, Param_3 := TRUE);`	`<Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="myInt" />`
	`  </Symbol>`
	`</Access>`
	`<Blank />`
	`<Token text=":=" />`
	`<Blank />`
	`<Access Scope="Call">`
	`  <CallInfo Name="MyFunction" BlockType="FC">`
	`  <Token text="(" />`
	`  <Parameter Name="Param_1">`
	`  ...`

Absolute call

In SCL, the call can be initiated using absolute address of the DB. Due to absolute address, the Name attriubute of CallInfo node is empty.

A recoverable exception is thrown by the import of

- An "Address" node is available, with valid value of Name attribute.
- Non-existence of Address node with no valid value of Name attribute.

SCL block	XML tag

`%DB20(...);`	`<Access Scope="Call">`
	`  <CallInfo Name="" BlockType="FB">`
	`  <Instance Scope="GlobalVariable">`
	`  <Address Area="DB" BlockNumber="20" />`
	`  </Instance>`
	`  <Token text="(" />`
	`  <Parameter>`
	`  …`
	`  </Parameter>`
	`  <Token text=")" />`
	`  </CallInfo>`
	`</Access>`

Instruction

The instruction in SCL block is checked in system library during the import operation, and the instruction versions are not exported in export operation.

The general instruction type is given below.

SCL block	XML tag

`#myInt := ATTACH(OB_NR := 1, EVENT := 15, ADD := TRUE);`	Format of XML exported in ExportOptions.ReadOnly setting
	
	`<Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="myInt" />`
	`  </Symbol>`
	`</Access>`
	`<Blank />`
	`<Token text=":=" />`
	`<Blank />`
	`<Access Scope="Call">`
	`  <Instruction Name="ATTACH">`
	`  <Token text="(" />`
	`  <Parameter Name="OB_NR">`
	`  <Blank />`
	`  <Token text=":=" />`
	`  <Blank />`
	`  <Access Scope="LiteralConstant">`
	`  <Constant>`
	`  <ConstantType>OB_ATT</ConstantType>`
	`  <ConstantValue>1</ConstantValue>`
	`  </Constant>`
	`  </Access>`
	`  </Parameter>`
	`  <Token text="," />`
	`  <Blank />`
	`  <Parameter Name="EVENT">`
	`  <Blank />`
	`  <Token text=":=" />`
	`  <Blank />`
	`  <Access Scope="LiteralConstant">`
	`  <Constant>`
	`  <ConstantType>EVENT_ATT</ConstantType>`
	`  <ConstantValue>15</ConstantValue>`
	`  </Constant>`
	`  </Access>`
	`  </Parameter>`
	`  <Token text="," />`
	`  <Blank />`
	`<Parameter Name="ADD">`
	`  <Blank />`
	`  <Token text=":=" />`
	`  <Blank />`
	`  <Access Scope="LiteralConstant">`
	`  <Constant>`
	`  <ConstantType>Bool</ConstantType>`
	`  <ConstantValue>TRUE</ConstantValue>`

SCL block	XML tag
	</Constant> </Access> </Parameter> <Parameter Name="RET_VAL" Informative="true" /> <Token text=")" /> </Instruction> </Access> <Token text=";" />

Instruction with template

When the template parameter is complements the instruction name, the export of the template parameter is necessary. If a "TemplateValue" tag with attribute Type="Type" follows the Instruction tag, the import operation concatenates the template value to the instruction name.

SCL block	XML tag
"tag_4" := MIN_DINT(IN1:="Tag_1", IN2:="Tag_2", IN3:="Tag_3");	<Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_4" /> </Symbol> </Access> ... <Access Scope="Call"> <Instruction Name="MIN"> <TemplateValue Name="value_type" Type="Type">DInt</TemplateValue> ... <Parameter Name="IN1"> ... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_1" /> </Symbol> </Access> </Parameter> ... <Parameter Name="IN2">... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_2" /> </Symbol> </Access> </Parameter> ... <Parameter Name="IN3"> ... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_3" /> </Symbol> </Access> </Parameter> ... </Instruction> </Access> ...

Conversion

For conversion functions, the real instruction name and its template values are not exported. Instead, the name used in the SCL block is exported.

SCL block	XML tag

`#output_1 :=`
`TIME_TO_S5TIME(#input_1);`
	`<Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component Name="output_1" />`
	`  </Symbol>`
	`</Access>`
	`...`
	`<Access Scope="Call">`
	`  <Instruction`
	`Name="TIME_TO_S5TIME">`
	`  <Token text="(" />`
	`  <NamelessParameter>`
	`  <Access Scope="LocalVariable">`
	`  <Symbol>`
	`  <Component`
	`Name="input_1" />`
	`  </Symbol>`
	`  </Access>`
	`  </NamelessParameter>`
	`  <Token text=")" />`
	`  </Instruction>`
	`</Access>`
	`...`

Instruction with instance

The instance and instruction are separated by blanks. Blanks are optional, and they can represented by new lines and comments. The instruction TON is represented by Name attribute of Instruction tag.

SCL block	XML tag

`IEC_Timer_0_DB . TON (IN:="Tag_1", PT:="Tag_2");`	`<Access Scope="GlobalAccess">`
	`  <Symbol>`
	`  <Component Name="IEC_Timer_0_DB" />`
	`  </Symbol >`
	`</Access>`
	`<Blank />`
	`<Token text="." />`
	`<Blank />`
	`<Access Scope="Call">`
	`  <Instruction Name="TON">`
	`  <Blank />`
	`  <Token text="(" />`
	`  <Parameter Name="IN">`
	`  <Access Scope="GlobalVariable">`
	`  <Symbol>`
	`  <Component Name="Tag_1" />`
	`  </Symbol>`
	`  </Access>`
	`  </Parameter>`
	`...`
	`  <Token text=")" />`
	`  </Instruction>`
	`</Access>`
	`<Token text=";" />`

Alarm constant

The alarm constants are only used in S7 400 PLCs, and the exported XML is similar to other languages.

SCL block	XML tag

`"Block_1_DB"(16#0000_0001);`	Format of XML exported in ExportOptions.None setting
	
	`<Access Scope="Call">`
	`  <CallInfo Name="Block_1" BlockType="FB">`
	`  <Instance Scope="GlobalVariable">`
	`  <Component Name="Block_1_DB" />`
	`  </Instance>`
	`  <NamelessParameter>`
	`  <Access Scope="AlarmConstant">`
	`  <Constant>`
	`  <ConstantType>C_Alarm_8</ConstantType>`
	
	`<ConstantValue>16#0000_0001</ConstantValue>`
	`  </Constant>`
	`  </Access>`
	`  </NamelessParameter >`
	`  </CallInfo>`
	`</Access>`

	Format of XML exported in ExportOptions.ReadOnly setting
	
	`<Access Scope="Call">`
	`  <CallInfo Name="Block_1" BlockType="FB">`
	`  <Instance Scope="GlobalVariable">`
	`  <Component Name="Block_1_DB" />`
	`  </Instance>`
	`  <NamelessParameter>`
	`  <Access Scope="AlarmConstant" >`
	`  <Constant>`
	`  <ConstantValue>16#00000001</ConstantValue>`
	`  <ConstantType>C_Alarm</ConstantType>`
	`  <StringAttribute Name="Format" Informative="true">Hex</StringAttribute>`
	`  </Constant>`
	`  </Access>`
	`  </NamelessParameter>`
	`  </CallInfo>`
	`</Access>`

ENO (Enable Output)

To support the ENO construct in SCL block, an attribute named "Scope" with value "PredefinedVariable" is used in the "Access" tag. It also contains the "PredefinedVariable" tag as child of the Access tag.

- The "PredefinedVariable" tag has one mandatory "Name" attribute.
- The scope "PredefinedVariable" and the tag "PredefinedVariable" are only allowed for SCL.

SCL block	XML tag

`Call(…, ENO => ENO);`	`<Access Scope="Call">`
	`<CallInfo BlockType="FC">`
	`<Token text="(" />`
	…
	`<Token text="," />`
	`<Blank />`
	`<Parameter Name="ENO">`
	`<Blank />`
	`<Token text="=>" />`
	`<Blank />`
	`<Access Scope="PredefinedVariable">`
	`<PredefinedVariable Name="ENO" />`
	`</Access>`
	`</Parameter>`
	`<Token text=")" />`
	`</CallInfo>`
	`</Access>`
	`<Token text=";" />`

`IF ENO = #c THEN …`	`<Token text="IF" />`
	`<Blank />`
	`<Access Scope="PredefinedVariable">`
	`<PredefinedVariable Name="ENO" />`
	`</Access>`
	`<Blank />`
	`<Token Text="=" />`
	`<Blank />`
	`<Access Scope="LocalVariable">`
	`<Symbol>`
	`<Component Name="c" />`
	`</Symbol>`
	`</Access>`
	`<Blank />`
	`<Token Text="THEN" />`

6.4.2.8 Export/Import multilingual comments in SCL

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

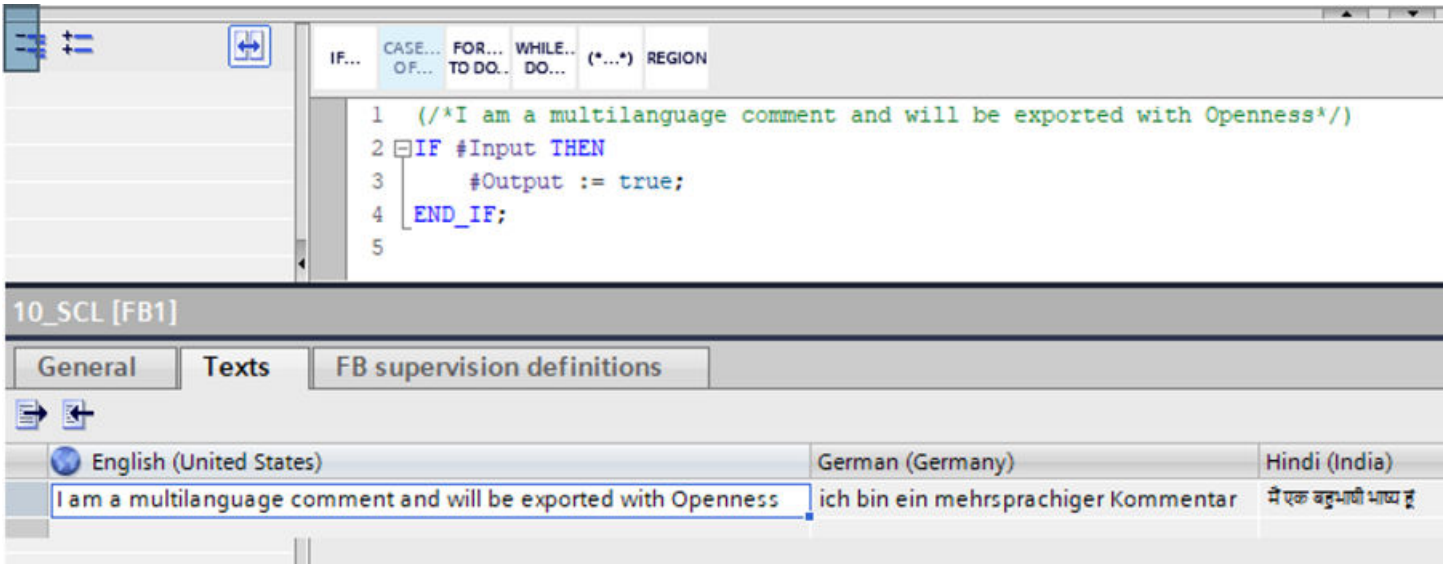
Application

It is possible in TIA Portal Openness to support import and export multilingual comments in SCL editor by the following ways:

- During SCL export in Openness, all the multilingual comments with its translation, according to enabled project languages, are exported into the same openness xml file.
- During SCL import in Openness, all the multilingual comments with its translation, according to enabled project languages, shall be imported back from the openness xml file. It shall be reflected in the project text area. The language gets enabled in if the language is not enabled in the project during import.
- After import, it shall be verified that all multilingual comment in SCL editor is liked to the appropriate comments in the project text.

Example

The following figure shows an example of Multilingual comment in SCL Editor:



The following figure shows an example for export of multilingual comment in openness XML file:

```
<Comment Inserted="true" UId="21">
  <MultiLanguageText Lang="en-US">I am a multilanguage comment and will be exported with Openness</MultiLanguageText>
  <MultiLanguageText Lang="de-DE">ich bin ein mehrsprachiger Kommentar</MultiLanguageText>
  <MultiLanguageText Lang="hi-IN">मैं एक बहुभाषी भाष्य हूं</MultiLanguageText>
</Comment>
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.4.2.9 Exporting failsafe blocks

Exporting failsafe blocks

It is now possible to import and export the Failsafe blocks, but F-system blocks cannot be exported.

Importing failsafe blocks

Consistent F-blocks can be exported and re-imported. They will be created as F-blocks.

It will be imported as standard block if you remove the prefixes "`F_`" from the value of all attributes "`ProgrammingLanguage`" .

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Exporting blocks (Page 1351)

Exporting blocks with know-how protection (Page 1309)

6.4.2.10 Exporting system blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The project contains a system block.
- The system block is not a F-block
- PLC is not online.

Introduction

Only visible system blocks are available in the blocks composition, for example no FB or FC blocks are available in the block composition . The resulting XML file is similar to the export file of a block.

Program code

Modify the following program code to export the visible data of a block to an XML file:

```
//Exports system blocks
private static void ExportSystemBlocks(PlcSoftware plcsoftware)
{
PlcSystemBlockGroup sbSystemGroup = plcsoftware.BlockGroup.SystemBlockGroups[0];
foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
{
foreach (PlcBlock block in group.Blocks)
{
block.Export(new FileInfo(string.Format(@"D:\Samples\{ 0 }.xml", block.Name)),
ExportOptions.WithDefaults);
}
}
```

6.4.2.11 Exporting GRAPH blocks with multi-language text

XML structure of GRAPH blocks with multi-language text

The export XML for GRAPH blocks contains the translated step names and transition names of the GRAPH. These translated multi-language text are represented as StepName and TransitionName elements under the parent element Step and Transition respectivelty. These elements contain one MultiLanguageText element for each supported language. The texts for the languages which are not set explicitly are not exported. If no translation is made, the StepName and TransitionName elements are not exported. The StepName and TransitionName elements are optional. The TIA Portal Openness XML import operation throws a recoverable exception for the graph versions < V5.0.

Example for StepName element

```
<Steps>
 <Step Number="1" Init="true" Name="Step1" MaximumStepTime="T#10S" WarningTime="T#7S">
   <StepName>
      <MultiLanguageText Lang="de-DE">stepDE</MultiLanguageText>
      <MultiLanguageText Lang="en-US">stepEN</MultiLanguageText>
      <MultiLanguageText Lang="it-CH">stepIT</MultiLanguageText>
   </StepName>
      ..
 </Step>
..
</Steps>
```

Example for TransitionName element

```
<Transitions>
  <Transition IsMissing="false" Name="Trans1" Number="1" ProgrammingLanguage="LAD">
   <TransitionName>
      <MultiLanguageText Lang="de-DE">transDE</MultiLanguageText>
      <MultiLanguageText Lang="en-US">transEN</MultiLanguageText>
      <MultiLanguageText Lang="it-CH">transIT</MultiLanguageText>
   </TransitionName>
      ..
 </Transition>
..
</Transitions>
```

6.4.2.12 Importing block

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Introduction

The TIA Portal Openness API supports the import of blocks with "LAD", "FBD", "GRAPH", "SCL" or "STL" programming languages from an XML file. The following block types are supported:

- Function blocks (FB)
- Functions (FC)
- Organization blocks (OB)
- Global data blocks (DB)

Note

Importing optimized data blocks

Optimized data blocks are only supported by CPUs as of S7-1200. If you import optimized data blocks into S7-300 or S7-400, an exception is thrown and the import fails.

Response to importing

The following rules apply when importing a block:

- The XML file can contain less data than the block in the project, e.g., fewer parameters.
- Redundant information, such as call information, must be identical in the project and in the XML file. Otherwise, an exception is thrown.
- The data in the XML file may be "inconsistent" regarding their ability to be compiled in the TIA Portal.
- Attributes with the attributes "ReadOnly=True" and "Informative=True" are not imported.
- Missing instance DBs are not created automatically.
- If no block number is specified in the xml file, the block number is assigned automatically.
- If the block is not existing in the project and no version information is specified in the xml file, the version "0.1" is assigned.

Program code

Modify the following program code:

```
//Import blocks
private static void ImportBlocks(PlcSoftware plcSoftware)
{
PlcBlockGroup blockGroup = plcSoftware.BlockGroup;
IList<PlcBlock> blocks = blockGroup.Blocks.Import(new FileInfo(@"D:\Blocks\myBlock.xml"),
ImportOptions.Override);
}
```

Modify the following program code:

```
//Import system blocks
private static void ImportSystemBlocks(PlcSoftware plcSoftware)
{
PlcBlockSystemGroup systemblockGroup =
plcSoftware.BlockGroup.SystemBlockGroups[0].Groups[0];
IList<PlcBlock> blocks = systemblockGroup.Blocks.Import(new
FileInfo(@"D:\Blocks\myBlock.xml"), ImportOptions.Override);
}
```

6.4.2.13 Exporting blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Introduction

The API interface supports exporting consistent blocks and user data types to an XML file.

The XML file receives the name of the block. The following block types are supported:

- Function blocks (FB)
- Functions (FC)
- Organization blocks (OB)
- Global data blocks (DB)

The following programming languages are supported:

- STL
- FBD
- LAD
- GRAPH
- SCL

Attributes applicable for all blocks

The following attributes are exported in all blocks with the selected `ExportOptions` (see Exporting configuration data (Page 1211)). Attributes in bold typeface are always exported.

Additional information is available in the TIA Portal information system under "Overview of block attributes".

Attribute	Type	Default value	ReadOnly
AutoNumber	Bool	true	false
CodeModifiedDate	DateTime	-	true
CompileDate	DateTime	-	true
CreationDate	DateTime	-	true
HeaderAuthor	String	""	false
HeaderFamily	String	""	false
HeaderName	String	""	false
HeaderVersion	String	"0.1"	false
Interface	String	empty interface	false
InterfaceModifiedDate	DateTime	-	true
IsConsistent	Bool	-	true
IsKnowHowProtected[1]	Bool	false	true
IsWriteProtected	Bool	false	true
MemoryLayout	enum MemoryLayout	-	false
ModifiedDate	DateTime	-	true
Name	String	-	false
Number	Int32	next available number	false
ParameterModified	DateTime	-	true
PLCSimAdvancedSupport	Bool	false	true
ProgrammingLanguage	enum ProgrammingLanguage	-	false
StructureModified	DateTime	-	true

[1] The IsKnowHowProtected attribute is applicable for UDT too.

Note

There are certain conditions where MemoryLayout attribute can be readonly.

Dynamic accessible general attribributes

The following attributes which can be read only in some conditions:

Attribute	ReadOnly condition
AutoNumber	All KnowHowProtected blocks, All Classic OBs; Plus OBs: DiagnosticErrorInterrupt, IOAccessError, ProgrammingError, PullOrPlugOfModules, RackOrStationFailure, Status, TimeErrorInterrupt, Update
HeaderVersion	System- and KnowHowProtected DBs; ArrayDBs that originated from system library
HeaderName	
HeaderFamily	
HeaderAuthor	
MemoryLayout	Classic blocks, System Know how protected blocks, ArrayDBs, IDBofFBs, Graph blocks

Attributes applicable for ArrayDB block

The following attributes are exported for ArrayDB block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
ArrayDataType	String	-	true
ArrayLimitUpperBound	Int32	-	true

Attributes applicable for DB block

The following attributes are exported in DB block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
IsOnlyStoredInLoadMemory	Bool	false	false
IsPLCDB	Bool	false	false
IsWriteProtectedInAS	Bool	false	false

Attributes applicable for FB block

The following attributes are exported for FB blocks with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
AssignedProDiagFB	String	-	-
ISMultiInstanceCapable	Bool	-	true
Supervisions	String	no supervisions	true for IDB of FB and false for FB

Attributes applicable for DB and FB blocks

The following attributes are exported in DB and FB block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
IsIECCheckEnabled	Bool	false	false
IsRetainMemResEnabled[1]	Bool	false	false
MemoryReserve	Unsigned	0	false
RetainMemoryReserve[2]	Unsigned	0	false

[2] If the "IsRetainMemResEnabled" attribute's value is "false", and the "RetainMemoryReserve" attribute is not equal to "0", an exception is thrown.

Attributes applicable for FB, DB and IDB blocks

The following attributes are exported in FB, DB, and IDB blocks with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
DownloadWithoutReinit	Bool	false	true

Attributes applicable for FB and FC blocks

The following attributes are exported for FB and FC block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
LibraryType	String	-	true
LibraryTypeVersionGuid	String	-	true

Attributes applicable for FB and FC (STL) blocks

The following attributes are exported for FB and FC (STL) block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
ParameterPassing	Bool	false	false

Attributes applicable for FB, FC and instance DB of an FB block

The following attributes are exported for FB, FC and instance DB of an FB block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
UDABlockProperties	String	""	false
UDAEnableTagReadback	Bool	false	false

Attributes applicable for instance DB of FB and UDT

The following attributes are exported for instance DB of FB and UDT block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
InstanceOfName	String	""	false
InstanceOfNumber	Unsigned Short	-	true
InstanceOfType	enum BlockType	-	true

Attribute	Type	Default value	ReadOnly
OfSystemLibElement	String	""	false
OfSystemLibVersion	String	""	false

Attributes applicable for OB block

The following attributes are exported in OB block for specific Plus PLCs with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
ApplicationCycle	Single	-	true
AutomaticMinimum	Bool	-	true
ConstantName	String	-	true
CycleTimeDistributedIO	Single	-	true
CyclicApplicationCycleTime	Single	-	true
CyclicTime	Int32	100000	true
DataExchangeMode	OBDataExchangeMode	Cyclic	true
DelayTime	Double	-	true
DistributedIOName	String	-	true
DistributedIONumber	Int32	-	true
EnableTimeError	Bool	-	true
EventClass	String	-	true
EventsToBeQueued	Int32	-	true
EventThresholdForTimeError	Int32	-	true
Execution	OBExecution	Never	true
Factor	Single	-	true
PhaseOffset	Int32	0	true
PriorityNumber	Int32	-	true
ProcessImagePartNumber	UInt32	-	true
ReportEvents	Bool	-	true
SecondaryType[3]	String	-	false
StartDate	DateTime	1/1/2012	true
SynchronousApplicationCycleTime	Single	-	true
TimeMode	OBTimeMode	System	true
TimeOfDay	DateTime	12:00 AM	true
TransformationDBNumber	UInt16	0xffff	true

[3] When exporting an OB, the "SecondaryType" is additionally set based on the OB number. The assignment is checked during import. If the assignment is incorrect, an exception of type "Recoverable" is thrown.

Attributes applicable for FB, FC and OB blocks

The following attributes are exported for FB, FC and OB block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
HandleErrorsWithinBlock	Bool	false	true

Attributes applicable for FB, FC and UDT blocks

The following attributes are exported for FB, FC and UDT block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
LibraryConformanceStatus	String	-	false

Attributes applicable for GRAPH block

The following attributes are exported for GRAPH block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
AcknowledgeErrorsRequired	Bool	true	false
CreateMinimizedDB	Bool	false	false
ExtensionBlockName	String	-	-
GraphVersion	String	-	false
InitialValuesAcquisition	String	-	-
LanguageInNetworks	String	-	false
LockOperatingMode	Bool	false	false
PermanentILProcessingIn-MANMode	Bool	false	false
SkipSteps	Bool	false	false

Attributes applicable for GRAPH FB block

The following attributes are exported for GRAPH FB block with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
WithAlarmHandling	Bool	true	false

Attributes applicable for SCL block

The following attributes are exported for SCL blocks with the selected `ExportOptions`. These attribues are exported based on the type of PLCs.

Attribute	Type	Default value	ReadOnly
CheckArrayLimits	Bool	false	false
ExtendedStatus	Bool	false	false
DBAccessibleFromOPCUA	Bool	true	false

Attributes applicable for GRAPH, SCL, and LAD/FBD blocks

The following attributes are exported for GRAPH, SCL, and LAD/FBD blocks with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
SetENOAutomatically	Bool	-	false

Attributes applicable for program blocks (except OB), DB, and UDT blocks

The following attributes are exported for program blocks (except OB), DB, and UDT blocks under a unit with the selected `ExportOptions`.

Attribute	Type	Default value	ReadOnly
Access	UnitAccessType	Unpublished	false

For Import behavior for the attribute 'Access'

	Imported under unit	Imported not under unit (without SWImportOptions.IgnoreUnitAttributes)	Imported not under unit (with SWImportOptions.IgnoreUnitAttributes)
XML exported from under unit	'Access' used and set	RecoverableException is thrown	'Access' ignored
XML exported from not under unit	'Access' gets its default value	'Access' doesn't exist	'Access' doesn't exist

Program code

Modify the following program code to export a block without know-how protection to an XML file:

```
//Exports a regular block
private static void ExportRegularBlock(PlcSoftware plcSoftware)
{
PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
plcBlock.Export(new FileInfo(string.Format(@"D:\Samples\{ 0 }.xml", plcBlock.Name)),
ExportOptions.WithDefaults);
}
```

6.4.2.14 Importing blocks/UDT with open reference

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- PLC is not online

Introduction

You can use the TIA Portal Openness to import blocks and UDTs even if a related object is missing. You can use the new mode by a new overload of the respective Import method. The new overload has an additional parameter which accepts a value of the new flagged enum SWImportOption

The Openness interface supports the new import mode for the following conditions:

Import of	Object reference
UDT	UDT
DB (global)	UDT
IDBofUDT	UDT
IDBofFB	FB
ArrayDB	Array of UDT
FB	UDT (interface), Multi-instance
FC	UDT (Interface)

Program code

To allow the import, you can use SWImportOptions.IgnoreMissingReferencedObject, even if the referenced object is missing.

```
[Flagged] Enum SWImportOptions
{
None = 0,
IgnoreStructuralChanges = 1,
IgnoreMissingReferencedObjects = 2
}
private static void Main(string[] args)
{
//... // All kinds of blocks
FileInfo file = ...;
PlcBlockComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreMissingReferencedObjects);
//...
//... // UDTs
PlcTypeComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreMissingReferencedObjects);
//...
}
```

See also

Opening a project (Page 128)

6.4.2.15 Importing blocks/UDT for structural change object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- PLC is not online

Application

Using Openness API, you can import blocks and UDTs even if instance data is lost because of a structural change of related objects.

The Openness interface supports the new import mode for the following conditions:

Import of	Object references
Tag	UDT
UDT	UDT

Import of	Object references
DB (global)	UDT
IDBofUDT	UDT
IDBofFB	FB
ArrayDB	Array of UDT
FB	UDT (interface), Multi-instance
FC	UDT (Interface)

Program code

You can use new mode by a new overload of the respective Import method. The new overload has an additional parameter which accepts a value of the new flagged enum SWImportOptions. To allow the import althoughthere are structural change and maybe a data loss using "SWImportOptions.IgnoreStructuralChanges".

```
Flagged Enum SWImportOptions
{
None = 0,
IgnoreStructuralChanges = 1,
IgnoreMissingReferencedObjects = 2
}
...
// All kinds of blocks
PlcBlockComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreStructuralChanges);
...
...
// UDTs
PlcTypeComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreStructuralChanges);
...
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.4.2.16 Export/Import of unit specific publishing attribute of blocks and types

Requirement

- The TIA Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

The Access attribute exists only for blocks, plc types, and tag tables located under a unit. The XML file of block, type, and tag table contains the Access attribute when exporting them into unit

environment and get its Unpublished default value while importing . The exported XML of the same block, type, and tag table does not contain the Access attribute in non-unit environment

and gets nothing while importing.

The general Openness XML import rules does not allow to import XMLs containing undefined attributes and undefined values. The rules restrict the export and import of XMLs originated from

object in a unit environment to a non-unit environment.

The rules add limitation in the usage of export and import of XML, therefore the import overloading method is defined which accepts the three parameters

Parameter	Return type	Description
path	String	Specifies the path to the Simatic ML file to be imported
importOptions	enum: Siemens.Engineering.ImportOptions	Specifies the general import option(s) to be used during import.
swImportOptions	enum: Siemens.Engineering.SW.SWImportOptions	Specifies the import option(s) specific to Step7 to be used during import.

The Enum type Siemens.Engineering.SW.SWImportOptions will be extended with the following new import option:

- IgnoreUnitAttributes: Specifies that the import process shouldn't be aborted in case of unit related attributes are present in the XML and the import is performed in a non-unit environment.

The 'IgnoreUnitAttributes' is considered when the XML

- Is exported from a unit
- Contains the 'Access' attribute
- Is imported into non-unit environment

If the exported XML does not contain the Access attribute and it is imported into a unit the new import option will not be considered by the import logic at all.

Program code

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
PlcSoftware plcSoftware = plcTarget.GetService<SoftwareContainer>() as PlcSoftware;
PlcUnit plcUnit1 = plcUnitProvider.UnitGroup.Units[0];
 //assuming Unit_1 is already existing
PlcUnit plcUnit2 = plcUnitProvider.UnitGroup.Units[1];
 //assuming Unit_2 is already existing
PlcBlock block1 = plcUnit1.BlockGroup.Blocks.Find("Block_1");
 //assuming Block_1 is already existing under Unit_1
PlcBlock block2 = plcUnit2.BlockGroup.Blocks.Find("Block_2");
 //assuming Block_2 is already existing under Unit_2
PlcBlock block3 = plcSoftware.BlockGroup.Blocks.Find("Block_3");
 //assuming Block_3 is already existing under the PLC
```

Program code: Object exported from a unit and imported into a unit

The examples below are demonstrating the behavior of the new import option by using blocks, the same applies for plc types and tag tables as well.

Modify the following program code to export a block with 'Access' attribute set to value 'Unpublished' (default value) with ExportOptions.WithDefaults and import with SWImportOptions.None:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);
```

Modify the following program code to export a block with 'Access' attribute set to value 'Unpublished' (default value) with ExportOptions.WithDefaults and import with SWImportOptions.IgnoreUnitAttributes:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.IgnoreUnitAttributes);
```

Modify the following program code to export a block with 'Access' attribute set to value 'Published' with ExportOptions.None and import with SWImportOptions.None:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo("somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);
```

Modify the following program code to export a block with 'Access' attribute set to value 'Published' with ExportOptions.None and import with SWImportOptions.IgnoreUnitAttributes:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo("somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.IgnoreUnitAttributes);
```

Note

In all of the above program codes, No exception is thrown, and import succeeds.

Program code: Object exported from a unit and imported into non-unit environment

Modify the following program code to export a block with 'Access' attribute set to value 'Unpublished' (default value) with ExportOptions.WithDefaults and import with SWImportOptions.None:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);
```

Note

In the above program code, Import does not succeed and a Recoverable Exception is thrown,

Modify the following program code to export a block with ‘Access’ attribute set to value ‘Unpublished’ (default value) with ExportOptions.WithDefaults and import with SWImportOptions.IgnoreUnitAttributes:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo(„somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);
```

Note

In the above program code, import succeed and No exception is thrown.

Modify the following program code to export a block with 'Access' attribute set to value 'Published' with ExportOptions.None and import with SWImportOptions.None:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo(„somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);
```

Note

In the above program code, Import succeed and a Recoverable exception is thrown.

Modify the following program code to export a block with ‘Access’ attribute set to value ‘Published’ with ExportOptions.None and import with SWImportOptions.IgnoreUnitAttributes:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo(„somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.IgnoreUnitAttributes);
```

Note

In the above program code, No exception is thrown and import succeed.

Object exported from non-unit environment into a unit

The exported XML doesn’t contain the ‘Access’ Openness attribute and when importing it gets the default value ‘Unpublished’.

Object exported from non-unit environment into non-unit environment

The exported XML doesn't contain the 'Access' Openness attribute and when importing it nothing happens.

6.4.2.17 Creating Instance DB

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is opening
  SeeOpening a project (Page 128)

Introduction

You can use the TIA Portal Openness to create instance DB for programming languages such as SCL, LAD, FBD, STL, Graph, and CEM.

Program code: Creating Instance DB for SCL, LAD, FBD, STL, Graph, & CEM block

```
PlcSoftware plc = ...;
// To create instance DB of SCL block
plc.BlockGroup.Blocks.CreateInstanceDB("ConveyerDB", true, 5, "Conveyer_SCL_Block");
// To create instance DB of LAD block
plc.BlockGroup.Blocks.CreateInstanceDB("RelayDB", true, 6, "Relay_LAD_Block");
// To create instance DB of FBD block
plc.BlockGroup.Blocks.CreateInstanceDB("SensorDB", true, 6, "Sensor_FBD_Block");
// To create instance DB of STL block
plc.BlockGroup.Blocks.CreateInstanceDB("ReadIODB", true, 6, "ReadIO_STL_Block");
// To create instance DB of Graph block
plc.BlockGroup.Blocks.CreateInstanceDB("ConveyerDB", true, 6, "Conveyer_Graph_Block");
// To create instance DB of CEM block
plc.BlockGroup.Blocks.CreateInstanceDB("RelayDB", true, 6, "Relay_CEM_Block");
```

6.4.2.18 Accessing DB value parameter without export

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- You have opened a project via a TIA Portal Openness application
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to read and write the following values of a member attribute in any DB.

Property name	Data type	Modelled/Dynamic	Access
Name	String	Modelled	Read
StartValue	String	Dynamic	Read/Write
AssignedPro-DiagFB	String	Dynamic	Read
ExternalAcces-sible	bool	Dynamic	Read/Write
ExternalVisible	bool	Dynamic	Read/Write
ExternalWrita-ble	bool	Dynamic	Read/Write
Retain	bool	Dynamic	Read/Write

Property name	Data type	Modelled/Dynamic	Access
SetPoint	bool	Dynamic	Read/Write
DataTypePoint	bool	Dynamic	Read/Write
Snapshot	bool	Dynamic	Read
DefaultValue	bool	Dynamic	Read

Program code

Modify the following program code to access the data block interface member details:

```
public static void PrintMemberAttributes( plcBlockInterface)
{
foreach(Member member in plcBlockInterface.Members)
{
Console.WriteLine($"Name: {member.Name}");
Console.WriteLine($"StartValue: {member.GetAttribute("StartValue")}");
Console.WriteLine($"ExternalAccessible: {member.GetAttribute("ExternalAccessible")}");
Console.WriteLine($"ExternalVisible: {member.GetAttribute("ExternalVisible")}");
Console.WriteLine($"ExternalWritable: {member.GetAttribute("ExternalWritable")}");
Console.WriteLine($"Retain: {member.GetAttribute("Retain")}");
Console.WriteLine($"Setpoint: {member.GetAttribute("Setpoint")}");
Console.WriteLine($"DataTypeName: {member.GetAttribute("DataTypeName")}");
Console.WriteLine($"Snapshot: {member.GetAttribute("Snapshot")}");
Console.WriteLine($"DefaultValue: {member.GetAttribute("DefaultValue")}");
}
}
```

Modify the following program code to read start value from DB:

```
PlcBlockInterface bi = dbbblock.Interface;
MemberComposition members = bi.Members;
Member member = members.Find("Room_Temperature");
string startValue = member.StartValue;
//Normal  get attribute  should  be  possible  as  usual
startValue = member.GetAttribute("StartValue");
//Array
initialSpeedvar = members.Find("Motor.InitialSpeed[0]");
object motorInitialSpeed = initialSpeedvar.StartValue;
//UDT-struct
axisSpeedvar = members.Find("FillingStation.Conveyer.AxisSpeed");
object axisSpeed = axisSpeedvar.StartValue;
axisSpeed = axisSpeedvar.GetAttribute("StartValue");
//Struct
initialSpeedvar = paramF.Find("DischargeValve.FlowMeter.InitialSpeed[0]");
object initialSpeed = initialSpeedvar.StartValue;
initialSpeed =  initialSpeedvar.GetAttribute("StartValue");
```

Modify the program code to write start value to DB:

```
PlcBlockInterface bi = dbbblock.Interface;
MemberComposition members = bi.Members
Member member = members.Find("Room_Temperature");
member.StartValue = 10.2;
//Normal  set attribute  should  be  possible  as  usual
member.SetAttribute("StartValue", 20.3);
//Array
initialSpeedvar = members.Find("Motor.InitialSpeed[0]");
initialSpeedvar.StartValue = 36;
initialSpeedvar.SetAttribute("StartValue", 56);
//UDT-struct
axisSpeedvar = members.Find("FillingStation.Conveyer.AxisSpeed");
object axisSpeed = axisSpeedvar.StartValue;
axisSpeed.SetAttribute("StartValue", 40.5);
initialSpeedvar = paramF.Find("DischargeValve.FlowMeter.InitialSpeed[0]");
object initialSpeed = initialSpeedvar.StartValue;
initialSpeed.SetAttribute("StartValue",  12);
```

Modify the following program code to set the start value of a tag to 'myTest':

```
PlcSoftware myPlcSoftware =
tiaProject.Devices[0].DeviceItems[1].GetService<SoftwareContainer>().Software as
PlcSoftware;
DataBlock myDatablock = myPlcSoftware.BlockGroup.Blocks[1] as DataBlock;
myDatablock.Interface.Members[0].SetAttribute("StartValue", "'myTest'");
```

See also

Opening a project (Page 128)

6.4.2.19 Export/Import of Plc Alarm TextLists

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to export and import Plc alarm text lists. The format of the Export/Import is XLSX.

Excel file structure

The generated XLSX file consists of two sheets, one for the text lists and other for the entries. The created XLSX file will receive a custom property named FileContent with value "Alarm text lists". If this property is missing or invalid (contains no or different value), the import will be denied.

The data is linked together between the two sheets. The TextListEntry.Parent refers to the TextList.Name

	A	B	C	D	E
1	Name	ListRange	Comment [en-US]	Comment [de-DE]	Comment [fr-FR]
2	NextTextList	Decimal	English	Deutsch	French

A	B	C	D	E	F
Parent	From	To	Text [de-DE]	Text [en-US]	Text [fr-FR]
NextTextList	0	1	xyzxyzxyz	rstrstrst	uvwuvwuvw

Engineering object model

The Engineering Object Model definition of the newly introduced PlcAlarmTextListProvider type. It is accessible both under a PlcUnit and PlcSoftware Engineering object.

Export

The Two API methods are provided for the export. If all text lists of a PLC/SW Unit in all activated project languages is to be exported, the only required parameter is the 'path' that defines the

location where the result XLSX file has to be located after the successful export.

```
FileInfo fileInfo = new FileInfo(Path.Combine(Environment.CurrentDirectory,
$"{xlsxName}.xlsx"));
PlcAlarmTextListProvider textListProvider = GetTextListProvider(unitName);
textListProvider.ExportToXlsx(fileInfo);
```

If not all text lists of a PLC/SW Unit has to be exported or the languages - in which the text lists have to be exported - should be filtered, another method is also available in the API, where the list

of text list names identify the text lists to be exported, and the list of Languages identify the languages in which the text lists have to be exported.

A sample usage of the ExportToXlsx action:

```
private void ExportPlcOrUnit(
string stationName,
string plcName,
string xlsxName,
string[] textListNames,
string[] cultureCodes,
string unitName = null)
{
LanguageSettings languageSettings = Project.LanguageSettings;
LanguageComposition supportedLanguages = languageSettings.Languages;
IEnumerable<Language> cultureInfos = null;
if (cultureCodes != null)
{
cultureInfos = cultureCodes.Select(i =>
supportedLanguages.Find(CultureInfo.GetCultureInfo(i)));
}
FileInfo fileInfo = new FileInfo(Path.Combine(Environment.CurrentDirectory,
$"{xlsxName}.xlsx"));
PlcAlarmTextListProvider textListProvider = GetTextListProvider(stationName, plcName,
unitName);
TextListXlsxResult result = null;
string exceptionMessage = string.Empty;
ExceptionMessageData? userExceptionMessageData = null;
try
{
result = textListProvider.ExportToXlsx(fileInfo, textListNames, cultureInfos);
}
catch (EngineeringTargetInvocationException e)
{
exceptionMessage = e.ToString();
userExceptionMessageData = e.DetailMessageData.FirstOrDefault();
}
}
private PlcAlarmTextListProvider GetTextListProvider(string stationName, string plcName,
string unitName = null)
{
DeviceComposition devices = Project.Devices;
Device device = devices.Where(station => station.Name == stationName).FirstOrDefault();
if (device == null)
{
throw new InvalidOperationException($"The requested '{stationName}' station is not found");
}
PlcSoftware plcTarget = (PlcSoftware)FindSoftwareTarget(device, plcName);
if (plcTarget == null)
{
throw new InvalidOperationException($"The requested '{plcName}' PLC SW is not found")
}
if (unitName != null)
{
PlcUnitProvider unitProvider = plcTarget.GetService<PlcUnitProvider>();
PlcUnit unit = unitProvider.UnitGroup.Units.Where(unit => unit.Name ==
unitName).FirstOrDefault();
if (unit == null)
{
throw new InvalidOperationException($"The requested '{unitName}' unit is not found");
```

```
}
return unit.GetService<PlcAlarmTextListProvider>();
}
return plcTarget.GetService<PlcAlarmTextListProvider>();
}
```

Error Handling:

- If the given language is not activated as a project language then the UserException will be thrown. For example, "Required language 'Konkani (India)' cannot be exported, because it is not used culture in the project. The valid languages are 'German (Germany)', 'English (United States)'"
- If there is no user text list exist then the UserException will be thrown (TextListNotFoundException) with the following message: "There is no text list on the following item: <PLCName>."
- If a given text list name does not exist, or exists but not a user text list, a UserException will be thrown. Such as "Text list User_1 is not found at PLC_1."
  "Text list SYSTEM_SDiag_CmpCpuName cannot be exported, because it is not user text list."

Import

The following Import action is present at the PLCAlarmTextListProvider class. With the importOptions parameter it can be clarified if the import should override existing text lists or not. If the

importOptions is set to ImportOptions.None and during the import an already existing text list should be updated, a UserException will be thrown. If the importOptions parameter is set to

ImportOptions.Override, then the already existing text lists will be updated during the import.

Existing text lists are identified by their name. New data (text list, entry) will be appended. Existing text list will not be deleted if there is not any corresponding data in the import file. if

the Excel file contains text list entries for an already existing text list, the already existing entries will be removed, and the entries given in the Excel file will be imported to the TIA project for that

specific text list (No merge for text list entries).

If invalid entries exist in the file, the text list containing the invalid entry will not be imported. (Examples for invalid entries: overlap, inconsistent with the containing text list data type.).

The Texts will be imported only in languages, which are active in the project. If the file contains more languages than the project, texts in extra languages will not be imported and a warning will

be logged. If the project contains more languages than the file, texts which have no corresponding data in the file will remain untouched (or empty, if import inserts new text lists or ranges).

No import can be done into a read-only project. A UserException is expected in case the user tries to import into a read-only project.

Import is a complex process, so several kinds of errors can occur. If the error is not critical, the import will be finished and the method returns with a TextListXlsxResult object that contains the

path to the log file (that contains the messages that are generated during the import), and has a State of TextListXlsxResultState.Warning.

If the error is critical a UserException will be thrown that contains the details about the error. If the error is fatal, a NonRecoverableException will be thrown.

```
<?xml version="1.0" standalone="no"?>
<?xml-stylesheet type='text/xsl' href='MassDataHandlerLogFile.xsl'?>
<LogFile
titleName="PLCAlarmTextLists_InvalidLanguage.xlsx__2019.06.07_13.46.45.070__Import_Log.xml
"
projectName="D:\TIA\dev\WM5_WinCC_HW_Work\binaries\Debug\x64\Tests\Siemens.Simatic.AlarmSe
rvices.Integration.Test.Openness\TextListXlsxFiles\PLCAlarmTextLists_InvalidLanguage.xlsx"
 typeText="Type" messageText="Message" timeText="Time">
<LogEntry type="Warning" dateTime="3:46:45 PM">
<Message>The language ID in column 'Comment [abcd-EF]' is missing or is invalid (sheet
'TextList'). The texts in this language are not imported.</Message>
</LogEntry>
<LogEntry type="Warning" dateTime="3:46:45 PM">
<Message>The language ID in column 'Text [abcd-EF]' is missing or is invalid (sheet
'TextListEntry'). The texts in this language are not imported.
</Message>
</LogEntry>
<LogEntry type="Information" dateTime="3:46:45 PM">
<Message>Import completed: 2 text lists with 9 entries.</Message>
</LogEntry>
</LogFile>
```

Note

- XLSX files generated during Export by TIA Portal V16 should have FileVersion custom property specified with value 1.
- XLSX files generated during Export by TIA Portal V16 should have FileContent custom property specified with value Alarm text lists.
- Log file is generated during Import at the TIA project's Logs folder

Results

The TextListXlsxResult object contains information about the result of the export or import. It contains a FileInfo named LogFilePath, which points to the logfile of the finished process. The State

(which is of type TextListXlsxResultState) will show the final state of the process, which can be OK, Warning or Error. If the result is Error, the process failed, if it is Warning, the process partially

succeeded.

6.4.2.20 Export Document information to SimaticML file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal to export PlcTag, PlcTagTable, and PlcUserConstant to SimaticML file with user specific document information. The document information exported to the Simatic ML file from the TIA Portal and imported from the SimaticML file to the TIA Portal is not relevant.

The new export allows you to either ignore full document information while exporting the Openness data types such as PlcTag, PlcTagTable and PlcUserConstant or configure what document information must be available in the SimaticML file.

DocumentInfoOptions

You can use the Siemens.Engineering.DocumentInfoOptions to specify the document information you need to export in simaticML file. The DocumentInfoOptions is a flag enum that allows you to select multiple configurations. Below are the configurations available for Export:.

Enum	Description
None	No document info is exported
ExportSetting	Export data with only ExportSetting document information, for example it includes the details of ExportOption used for Export.
InstalledProducts	Export data with only InstalledProducts document information, for example it includes product names and option package names
CreatedTimeStamp	Export data with only CreatedTimeStamp document information
All	Export data with all document information (CreatedTimeStamp, ExportSetting, InstalledProducts)

Program code: Export with DocumentInfoOptions

```
PlcTagTable tagTable = FindTagTableToBeExported("TestTagTable");
FileInfo testTagTableExportFile = new FileInfo($@"E:\temp\TestTagTable.xml");
// No document info will be exported since the configuration is None.
tagTable.Export(testTagTableExportFile, ExportOptions.WithDefaults,
DocumentInfoOptions.None);
FileInfo testTagExportFile = new FileInfo($@"E:\temp\TestTag.xml");
PlcTag tag = tagTable.Tags.Find("TestTag");
// CreatedTimeStamp and InstalledProduct will be exported as DocumentInfo.
tag.Export(testTagExportFile, ExportOptions.WithDefaults,
DocumentInfoOptions.CreatedTimeStamp | DocumentInfoOptions.InstalledProducts);
FileInfo testUserConstantExportFile = new FileInfo($@"E:\temp\TestUserConstant.xml");
PlcUserConstant plcUserConstant = tagTable.UserConstants.Find("TestUserConstant");
// Full Document info will be exported
plcUserConstant.Export(testUserConstantExportFile, ExportOptions.WithDefaults,
DocumentInfoOptions.All);
```

6.4.2.21 Export/Import of Alarm Instance Text

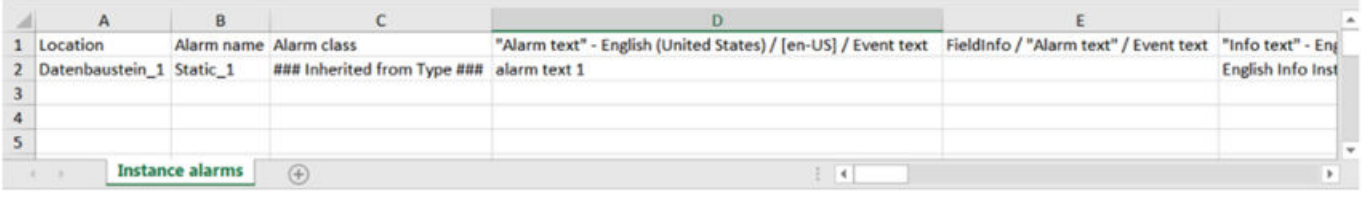
Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to export and import alarm instance texts which is similar to the export and import of type alarm text form PLC supervisions and alarms editor and also similar to the export/import of instance alarm texts from the CFC alarm editor. The export and import of alarm instance text function via TIA Portal Openness is intended to use for translation purposes. The function does not create any type of alarm instance and any alarm class during import. For example, if the alarm class is identified by name of the excel file exists in the target project, it will be set for the instance alarm during import but non-existing class will not be created.

Excel file structure



	A	B	C	D	E	
1	Location	Alarm name	Alarm class	"Alarm text" - English (United States) / [en-US] / Event text	FieldInfo / "Alarm text" / Event text	"Info text" - Eng
2	Datenbaustein_1	Static_1	### Inherited from Type ###	alarm text 1		English Info Inst
3						
4						
5						

Instance alarms

Inherited from type### is a special content which instructs the system to use the text or the Alarm class of the type. The Alarm class field refers to an existing alarm class name in the target project. Important rule that for a particular alarm text, either all or no language fields

should contain ###Inherited from type###. If there are some mixed content fields, no texts for that alarm will be imported at all and a warning will be issued.

Export function takes care about the consistency of the file data.

Freely over-writable fields are the language-dependent ones, i.e. fields which headers contain language identifiers, e.g. "Alarm text" - English (United States) / [en-US] / Event text.

The file also contains some special properties to let the system make some checks prior to import. If these properties are invalid, then import cannot be made.

Export

The following action is provided for export at the PlcAlarmTextProvider.

- PlcAlarmTextProvider

The following parameters are provided for import at the PlcAlarmTextProvider:

- Path
- Language
- Option

The export can be a PLC or a Software Unit. Path parameter is mandatory, Languages can be empty or null. In this case texts in all active project languages will be exported. Option parameter is also mandatory. It takes value(s) of PlcAlarmTextXlsxExportOption enum.

PlcAlarmTextXlsxExportOption has the following values:

- None (Only Alarm texts will be exported)
- IncludeInfoText (Also Info texts will be exported)
- IncludeAdditionalTexts (Also additional text 1 - 9 will contained in the export)
- IncludeAlarmClass (Alarm class setting willl be added to the export)
- All

The above three Include options can be combined, for example. if you want Alarm text, Additional texts and Alarm class but not the Info text, then we set IncludeAdditionalTexts and IncludeAlarmClass.

If an error occurs, UserException will be thrown. These cases could be, for example, inactive or non existing languages or file IO errors. Message for invalid language will be in the following format:

"Required language 'Konkani (India)' cannot be exported, because it is not used culture in this project. Valid languages are 'German (Germany)', 'English (United States)'"

Import

The following action is provided for import at the PlcAlarmTextProvider:

- ImportInstanceTextsToXlsx

The following parameters are provided for import at the PlcAlarmTextProvider:

- Path
- Language

The import can be a PLC or a Software Unit. Path parameter is mandatory, Languages can be empty or null. In this case texts in all active project languages will be imported. (If the excel file contains languages which are inactive in the project, those languages will not be activated and no import will be done in those languages.)

```
FileInfo fileInfo = new FileInfo(Path.Combine(s_XlsxFilesFolderName, $"{xlsxName}.xlsx"));
List<Language> cultureInfos = new List<Language>();
PlcSoftware plcTarget = ...;
PlcAlarmTextProvider alarmTextsProvider = plcTarget.GetService<PlcAlarmTextProvider>();
PlcAlarmTextXlsxResult result = alarmTextsProvider.ExportInstanceTextsToXlsx(fileInfo,
cultureInfos, PlcAlarmTextXlsxExportOption.All);
//Project project = ...;
LanguageSettings languageSettings = project.LanguageSettings;
LanguageComposition supportedLanguages = languageSettings.Languages;
cultureInfo = cultureInfos.Add(supportedLanguages.Find(CultureInfo.GetCultureInfo("en-
US")));
PlcAlarmTextProvider alarmTextsProvider = plcTarget.GetService<PlcAlarmTextProvider>();
result = alarmTextsProvider.ImportInstanceTextsFromXlsx(fileInfo, cultureInfo);
```

Result

Information about the result of the export or import will be provided in a PlcAlarmTextXlsxResult object. It contains a FileInfo structure as LogFilePath, which points to the logfile of the finished process. The State (which is of type PlcAlarmTextXlsxResultState) will show the final state of the process. Values can be OK, Warning or Error. If the result is Error, the process failed, if it is Warning, the process partially succeeded.

Exception

Non recoverable exceptions

These are technical issues, none of them should appear for you.

- "Missing PlcAlarmTextProvider"
- "Plc/SW Unit cannot be retrieved"
- Parameter has not found. Please contact Siemens technical support.
- "ProjectService is not available"
- path Parameter has not found. (Parameters: nnn, mmm) Please contact Siemens technical support.
- Result State is not error, but there are some error messages: xyz.

Recoverable exceptions

These could appear to you due to e.g improper parameter values.

- Required language 'xx-XX' cannot be exported, because it is not set for this PLC. Valid languages are yy-YY, zz-ZZ.

Various system error messages can also appear, e.g. file IO errors, etc.

See also

Opening a project (Page 128)

6.4.2.22 Export/Import of Alarm classes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to export/import Alarm classes. The format of the Export/Import is .DAT.

Engineering Object Model

The Engineering object model definition of the newly introduced AlarmClassDataProvider type.

It is accessible under the ProjectBase Engineering object.

Export

The following action is present at the AlarmClassDataProvider class for Export:

The export action has one parameter: path.

With the path parameter the file path can be given where the result of the export is to be placed. The type of the parameter is FileInfo.

The result of the action is an AlarmClassExportImportResult object. It contains the count of errors, warnings and the state of the export result (Success, Warning or Error).

It also has a Messages composition property, that contains the messages generated during the export together with the state (Success, Error or Warning).

A sample usage of the Export action:

```
FileInfo fileInfo = new FileInfo(@"D:\AlarmClasses.DAT");
AlarmClassDataProvider provider = Project.GetService<AlarmClassDataProvider>();
if (fileInfo.Exists)
{
fileInfo.Delete();
}
AlarmClassExportImportResult result;
result = provider.Export(fileInfo);
Assert.AreEqual(AlarmClassExportImportResultState.Success, result.State);
Assert.AreEqual(0, result.ErrorCount);
Assert.AreEqual(0, result.WarningCount);
AlarmClassExportImportResultMessageComposition messages = result.Messages;
Assert.AreEqual(1, messages.Count);
Assert.AreEqual($"Export of Alarm class settings file '{fileInfo.FullName}' was
successful.", messages.FirstOrDefault().Message);
Assert.AreEqual(AlarmClassExportImportResultState.Success,
messages.FirstOrDefault().State);
Assert.IsTrue(File.Exists(fileInfo.FullName), "Exported file does not exist.");
fileInfo.Delete();
```

Error Handling:

- If the overall state of the export process is Error a UserException is raised, and the list of error messages is stored in the exception itself.

From client side an EngineeringTargetInvocationException exception can be caught, where the DetailMessageData property contains the error messages.

- Possible errors:
  - IO error during the export process

Import

The import action is similar to the export action.

A sample of usage of the Import action:

```
FileInfo fileInfo = new FileInfo(@"D:\AlarmClasses.DAT");
AlarmClassDataProvider provider = Project.GetService<AlarmClassDataProvider>();
AlarmClassExportImportResult result;
result = provider.Import(fileInfo);
Assert.AreEqual(AlarmClassExportImportResultState.Success, result.State);
Assert.AreEqual(0, result.ErrorCount);
Assert.AreEqual(0, result.WarningCount);
AlarmClassExportImportResultMessageComposition messages = result.Messages;
Assert.AreEqual(1, messages.Count);
Assert.AreEqual($"Import of Alarm class settings file '{fileInfo.FullName}' was
successful.", messages.FirstOrDefault().Message);
Assert.AreEqual(AlarmClassExportImportResultState.Success,
messages.FirstOrDefault().State);
```

Error handling:

- If the overall state of the export process is Error a UserException is raised, and the list of error messages is stored in the exception itself.

From client side an EngineeringTargetInvocationException exception can be caught, where the DetailMessageData property contains the error messages

- Possible errors
  - File to be imported has invalid file extension
  - File to be imported missing file extension
  - With the import the count of alarm classes would exceed the maximum allowed number of alarm classes in the project
  - The file to be imported uses not the correct schema version (Only version 1.0 and 2.0 is supported)
  - No overlapping between the activated project languages and the languages given in the import file
  - No language is defined
  - IO error during the export process

Note

It shall be supported to import files generated during Export with TIA Portal V16.

Result

The AlarmClassExportImportResult object will contain an Information about the result of the export or import. It contains an AlarmClassExportImportResultMessageComposition navigator named Messages, which points to the messages generated during the export and import process. The State (which is of type AlarmClassExportImportResultState) will show the final state of the process, which can be Success, Warning or Error. If the result is Error, the process failed, if it is Warning, the process succeeded.

See also

Opening a project (Page 128)

6.4.2.23 Export/Import of global supervision for ProDiag-FB

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Opening a project (Page 128)
- A project is open
  See Opening a project (Page 82)

Introduction

You can use the TIA Portal Openness to support export/import of global supervisions of ProDiag block.. This functionality is supported by service provider mechanism and the same service provider is used by the block under Plc and software unit to perform export and import operations.

On perfoming export\import you shall be communicated with proper result state or with proper user exceptions.

The following method can be used to export and import global supervisions of ProDiag block:

Method	Description
ExportSupervisionsToXlsx(FileInfo path)	Export the supervisions tags including settings in the TIA portal supported format and generates the results of the export
ImportSupervisionsFromXlsx(FileInfo path, ImportOptions importOptions)	Import supervision tags with supported import options
ImportSupervisionSettingsFromXlsx(FileInfo path,ImportOptions importOptions)	Import supervision settings which are placed in exported file along with supervised tags

The following properties are available under SupervisionXlsxResult:

Property Name	Data Type	Access
LogFilePath	System.IO.FileInfo	Read
State	SupervisionXlsxResult	Read

The SupervisionXlsxResultState has the following enum values:

ENUM	Values
SupervisionXlsxResultState	Success
	Failure

Program code

Modify the following program code to export global supervisions of ProDiag block:

```
// File Path for the export
FileInfo fileInfo = new FileInfo(@"C:\Users\z003jwfc\Desktop\Supervisions_Openness.Xlsx");
//SW is nothing but PlcSoftware / PlcUnit.
var proDiagBlock = (FB)SW.BlockGroup.Blocks.Find("Block1");
SupervisionProvider supervisionProvider = proDiagBlock.GetService<SupervisionProvider>();
SupervisionXlsxResult result = supervisionProvider.ExportSupervisionsToXlsx(fileInfo);
```

Modify the following program code to import global supervisions of ProDiag block:

```
// File Path for the import
FileInfo fileInfo = new FileInfo(@"C:\Users\z003jwfc\Desktop\SupervisionsOpenness.Xlsx");
//SW is nothing but PlcSoftware / PlcUnit.
var proDiagBlock = (FB)SW.BlockGroup.Blocks.Find("Block1");
SupervisionProvider supervisionProvider = proDiagBlock.GetService<SupervisionProvider>();
//import supervisions
SupervisionXlsxResult result = supervisionProvider.ImportSupervisionsFromX1sx(fileInfo,
ImportOptions.None);
SupervisionXlsxResult result = supervisionProvider.ImportSupervisionsFromX1sx(fileInfo,
ImportOptions.Override);
//import supervision settings
SupervisionXlsxResult result =
supervisionProvider.ImportSupervisionSettingsFromX1sx(fileInfo, ImportOptions.None);
SupervisionXlsxResult result =
supervisionProvider.ImportSupervisionSettingsFromX1sx(fileInfo, ImportOptions.Override);
```

6.4.2.24 Export/import of ProDiag supervisions from Global overview editor

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- Opening a project
  See Opening a Project (Page 128)

Introduction

You can use the TIA Portal Openness to support export and import of global supervisions available under Plc or under Unit. The functionality is supported by service provider mechanism and the same service provider is used mainly by both Plc and software unit to perform export and import operations. On performing export\import, you shall be communicated with proper result or with exceptions.

The following method can be used to export and import global supervisions:

Method	Description
ExportSupervisionsToXlsx(FileInfo path)	Export the supervisions tags including settings in the TIA portal supported format and generates the results of the export
ImportSupervisionsFromXlsx(FileInfo path, ImportOptions importOptions)	Import supervision tags with supported import options
ImportSupervisionSettings-FromXlsx(FileInfo path,ImportOptions importOptions)	Import supervision settings which are placed in exported file along with supervised tags

The following properties are available under SupervisionXlsxResult:

Property name	Data type	Access
LogFilePath	System.IO.FileInfo	Read
State	SupervisionXlsxResult	Read

The SupervisionXlsxResultState has the following ENUM values:

ENUM	Values
SupervisionXlsxResultState	Success
	Failure

Introduction

Modify the following program code to export global supervisions:

```
//File Path for the export
FileInfo fileInfo = new FileInfo(@"C:\Users\z003jwfc\Desktop\Supervisions_Openness.Xlsx");
//SW is nothing but PlcSoftware / PlcUnit.
SupervisionProvider supervisionProvider = SW.GetService<SupervisionProvider>();
SupervisionXlsxResult result = supervisionProvider.ExportSupervisionsToXlsx(fileInfo);
```

Modify the following program code to import global supervisions:

```
//File Path for the import.
FileInfo fileInfo = new FileInfo(@"C:\Users\z003jwfc\Desktop\Supervisions_Openness.Xlsx");
//SW is nothing but PlcSoftware / PlcUnit.
Supervision Provider supervisionProvider = SW.GetService<SupervisionProvider>();
//import supervisions
SupervisionXlsxResult result = supervisionProvider.ImportSupervisionsFromXlsx(fileInfo,
ImportOptions.None);
SupervisionXlsxResult result = supervisionProvider.ImportSupervisionsFromXlsx(fileInfo,
ImportOptions.Override);
//import supervision settings
SupervisionXlsxResult result =
supervisionProvider.ImportSupervisionsSettingsFromXlsx(fileInfo, ImportOptions.None);
SupervisionXlsxResult result =
supervisionProvider.ImportSupervisionsSettingsFromXlsx(fileInfo, ImportOptions.Override);
```

See also

Opening a project (Page 128)

6.4.2.25 Export/Import of settings for ProDiag supervisions

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to support export/import of supervision settings.

The following properties are available under SupervisionSettingsExportImportResult:

Property Name	Data Type	Access
State	SupervisionSettingExportImportResult-State	Read
ErrorCount	Int32	Read
WarningCount	Int32	Read
Message	String	Read

The ENUM SupervisionSettingExportImportResultState has following value:

ENUM	Value
SupervisionSettingExportImportResultState	Success
	ErrorRollback

Program code

Modify the following program code to export supervision settings:

```
// MyProject obtains SupervisionSettingsProvider
SupervisionSettingsProvider settingsProvider =
MyProject.GetService<SupervisionSettingsProvider>();
// Import file path defined
FileInfo exportFile = new FileInfo(@"D:\Temp\ProDiagSettings.dat");
// Stores result after import
SupervisionSettingsExportImportResult exportResult = settingsProvider.Export(exportFile);
// Result state after import
SupervisionSettingsExportImportResultState resultState = exportResult.State;
// Total error count of result messages
int totalErrorCount = exportResult.ErrorCount;
// Composition of result messages
SupervisionSettingsExportImportResultMessageComposition exportMessageList =
exportResult.Messages;
// Count of result messages
exportMessageList.Count;
// Read out specific message text from each message
exportMessageList [0].Message;
// Output: "Export of file 'D:\temp\SupervisionSettings.dat' with the ProDiag supervision
settings was successful.";
```

Modify the following program code to import supervision settings:

```
// MyProject obtains SupervisionSettingsProvider
SupervisionSettingsProvider settingsProvider =
MyProject.GetService<SupervisionSettingsProvider>();
// Import file path defined
FileInfo importFile = new FileInfo(@"D:\Temp\ProDiagSettings.dat");
// Stores result after import
SupervisionSettingsExportImportResult importResult = settingsProvider.Import(importFile);
// Result state after import
SupervisionSettingsExportImportResultState resultState = importResult.State;
// Total error count of result messages
int totalErrorCount = importResult.ErrorCount;
// Composition of result messages
SupervisionSettingsExportImportResultMessageComposition importMessageList =
importResult.Messages;
// Count of result messages
importMessageList.Count;
// Read out specific message text from each message
importMessageList[0].Message;
// Output: "Import of the file 'D:\temp\SupervisionSettings.dat' with the ProDiag
supervision settings was successful.";
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.4.2.26 Export/Import Watch & Force Table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- You have opened a project with your TIA Portal Openness application.
  See Opening a project (Page 128)

Introduction

You can use the TIA Portal Openness to export Watch Table and Force Table from TIA Portal to SIMATIC ML and then import Watch Table and Force Table from SIMATIC ML.

The Export option in Watch Table can be set with the following definition (None, WithDefaults, WithReadOnly, WithDefaultsAndReadOnly). A WatchTable has only one published property i.e. the name. The name is published for read.

The imported WatchTable will be added to the list of WatchTables. The Import options in Watch Table should be set to the required one (None, Override). The ForceTables can be imported in a similar way, but it is only allowed to have one ForceTable.

If you use the None option instead of Override and a (non-empty) ForceTable already exists, then the import will fail with the RecoverableException: Only one ForceTable is allowed to be imported.

Program code: Export Watch Table and Force Table

Modify the following program code to export Watch Table:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Software as PlcSoftware;
PlcWatchTableComposition exportWatchTables =
plcSoftware.PlcWatchAndForceTableGroup.WatchTables;
PlcWatchTable watchTable = exportWatchTables.Find(watchTableName);
if(watchTable != null)
{
 watchTable.Export((FileInfo) fileInfo, ExportOptions.None);
}
```

Note

Export options in Watch Table should be set (None, WithDefaults, WithReadOnly, WithDefaultsAndReadOnly.). A WatchTable has only one published property i.e. the name. The name is published for read.

Modify the following program code to export Force Table:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Software as PlcSoftware;
PlcForceTableComposition exportForceTables =
plcSoftware.PlcWatchAndForceTableGroup.ForceTables;
PlcForceTable forceTable = exportForceTables[0];
forceTable.Export((FileInfo) fileInfo, ExportOptions.None);
```

Note

There is only one ForceTable in every situation, and it has a read-only name.

Program code: Import Watch Table and Force Table

Modify the following program code to import Watch Table:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Software as PlcSoftware;
PlcWatchTableComposition importWatchTables =
plcSoftware.PlcWatchAndForceTableGroup.WatchTables;
IList<PlcWatchTable> WatchTables = importWatchTables.Import((FileInfo)fileInfo,
ImportOptions.None);
```

Note

The imported WatchTable will be added to the list of WatchTables. Import options in Watch Table should be set to the required one (None, Override).

ForceTables can be imported in a similar way, but it is only allowed to have one ForceTable. If you use the None option instead of Override and a (non-empty) ForceTable already exists, then the import will fail with the following RecoverableException: Only one ForceTable is allowed to be imported. Please use Override importOption to overwrite the existing one.

6.4.2.27 Exporting user data type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online

Program code

Modify the following program code to export an user data type to an XML file:

```
//Exports a user defined type
private static void ExportUserDefinedType(PlcSoftware plcSoftware)
{
    string udtname = "udt name XYZ";
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find(udtname);
    udt.Export(new FileInfo(string.Format(@"C:\OpennessSamples\udts\{0}.xml", udt.Name)),
ExportOptions.WithDefaults);
}
```

6.4.2.28 Importing user data type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  Connecting to the TIA Portal (Page 82)
- A project is open.
  Opening a project (Page 128)
- PLC is not online.

Introduction

The API interface supports the importing of user data types from an XML file.

Import file syntax

The following code example shows an excerpt from an import file of a user-defined data type:

```
<Section Name="Input">
 <Member Name="Input1" Datatype=quot;myudt1&quot;>
  <Sections>
   <Section Name="None">
    <Member Name="MyUDT1Member1" Datatype="bool"/>
    <Member Name="MyUDT1Member2" Datatype=&quot;myudt1&quot;>
     <Sections...
```

Note

Syntax for user-defined data types of elements

An exception is thrown if the user-defined data type of an element in the import file for user data types has incorrect syntax.

Make sure that user-defined data types are noted with `&quot;`.

Program code

Modify the following program code to import a user data type:

```
//Imports user data type
private static void ImportUserDataType(PlcSoftware plcSoftware)
{
    FileInfo fullFilePath = new FileInfo(@"C:\OpennessSamples\Import\ExportedPlcType.xml");
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    IList<PlcType> importedTypes = types.Import(fullFilePath, ImportOptions.Override);
}
```

See also

Importing configuration data (Page 1212)

6.4.2.29 Export of data in OPC UA XML format

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online

Application

You can use the TIA Portal Openness to export PLC data as OPC UA XML file. As input parameter for the action you need an absolute directory path, where the xml file will be saved.

Program code

Modify the following program code to export PLC data as OPC UA XML file:

```
//Export PLC data as OPC UA XML file
private static void OpcUaExport(Project project, DeviceItem plc)
{
OpcUaExportProvider opcUaExportProvider = project.HwUtilities.Find("OPCUAExportProvider")
as OpcUaExportProvider;
if (opcUaExportProvider == null) return;
opcUaExportProvider.Export(plc, new FileInfo(string.Format(@"D:\OPC UA export
files\{0}.xml", plc.Name)));
}
```

6.4.2.30 Export/Import of UDT & DB

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal Openness to define a supervision for bool members inside UDT and assign prodiagFB for UDT instances in global data while creating the global Datablock. You should be able to provide and fetch this supervison information through TIA Portal Openness export and import respectively.

XML structure of exported/imported UDT

Use the below XML structure to export a element of boolean variable for which you have defined a supervision:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
<Engineering version="V17" />
<DocumentInfo>
<Created>2020-06-08T20:10:12.6308242Z</Created>
<ExportSetting>None</ExportSetting>
<InstalledProducts>
<Product>
<DisplayName>Totally Integrated Automation Portal</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>TIA Portal Openness</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
<OptionPackage>
<DisplayName>TIA Portal Version Control Interface</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
<Product>
<DisplayName>Feature Cycle 3 TIA Portal</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<Product>
<DisplayName>STEP 7 Professional</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>SIMATIC Energy Suite</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
<OptionPackage>
<DisplayName>STEP 7 Safety</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
<Product>
<DisplayName>WinCC Professional</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
</InstalledProducts>
</DocumentInfo>
<SW.Types.PlcStruct ID="0">
<AttributeList>
<Interface><Sections>
<Section Name="None">
<Member Name="Element_1" Datatype="Bool" />
</Section>
</Sections>
</Interface>
<Name>User_data_type_1</Name>
<Supervisions><PLCDataTypeSupervisions>
<PLCDataTypeSupervision Number="1" Type="Operand">
<SupervisedOperand Name="#Element_1" />
<SupervisedStatus>false</SupervisedStatus>
<DelayOperand Name="T#0ms" />
```

```
<Conditions>
<Condition>
<ConditionOperand Number="1" Name="" />
<TriggeringStatus>true</TriggeringStatus>
</Condition>
<Condition>
<ConditionOperand Number="2" Name="" />
<TriggeringStatus>true</TriggeringStatus>
</Condition>
<Condition>
<ConditionOperand Number="3" Name="" />
<TriggeringStatus>true</TriggeringStatus>
</Condition>
</Conditions>
<CategoryNumber>1</CategoryNumber>
<SubCategory1Number>0</SubCategory1Number>
<SubCategory2Number>0</SubCategory2Number>
</PLCDataTypeSupervision></PLCDataTypeSupervisions></Supervisions>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="3" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="4" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Types.PlcStruct>
</Document>
```

When you want to import a UDT and assign the supervision for the bool variable, the above-mentioned XML tags can be used. However, you cannot set the supervision information to a bool variable through openness after import.

Note

Supervision is only applicable for bool variables for UDT, when you attempt to import or export the "PLCDataTypeSupervisions" tags for any non-boolean members in UDT then no exception is thrown.

XML structure of exported /imported DB

Use the following XML structure to export a member of UDT instance irrespective of whether the UDT has supervision or not and an array of struct which has UDT instance as its member:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
<Engineering version="V17" />
<DocumentInfo>
<Created>2020-06-09T16:01:18.494539Z</Created>
<ExportSetting>None</ExportSetting>
<InstalledProducts>
<Product>
<DisplayName>Totally Integrated Automation Portal</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<Product>
<DisplayName>Feature Cycle 1 TIA Portal</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<Product>
<DisplayName>Feature Cycle 3 TIA Portal</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<Product>
<DisplayName>STEP 7 Professional</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>STEP 7 Safety</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
</InstalledProducts>
</DocumentInfo>
<SW.Blocks.GlobalDB ID="0">
<AttributeList>
<Interface><Sections>
<Section Name="Static">
<Member Name="Static_1" Datatype="Udt_With_Supervision"><AttributeList><BooleanAttribute
Name="SetPoint" SystemDefined="true">true</BooleanAttribute></
AttributeList><AssignedProDiagFB>Default_SupervisionFB</AssignedProDiagFB></Member>
<Member Name="Static_2" Datatype="Array[0..1] of Struct">
<Member Name="Static_1"
Datatype="&amp;quot;Udt_With_Supervision&amp;quot;"><AttributeList><BooleanAttribute
Name="SetPoint" SystemDefined="true">true</BooleanAttribute></AttributeList><Subelement
Path="0"><AssignedProDiagFB>Default_SupervisionFB</AssignedProDiagFB></
Subelement><Subelement Path="1"><AssignedProDiagFB>Default_SupervisionFB</
AssignedProDiagFB></Subelement></Member>
</Member>
</Section>
</Sections></Interface>
<MemoryLayout>Optimized</MemoryLayout>
<MemoryReserve>100</MemoryReserve>
<Name>Data_block_5</Name>
<Number>16</Number>
<ProgrammingLanguage>DB</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
```

```
<ObjectList>
<MultilingualTextItem ID="2" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="3" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="4" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.GlobalDB>
</Document>
```

Note

It is valid to have assignedProDiagFB attribute only for UDT instance. It is not possible to export any invalid data as it should be compile clean so you don't get any exception during export. You will encounter the not supported exception , if you try to import the "AssignedProDiagFB" tag for non UDT instances inside global DB or any other member in any block.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.4.2.31 Export/Import of Array & Instance DB

Quintessence

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to export and import the Array DB and instance DB of UDT instances so that the information about assigned proDiagFB sould be exported appropriately.

Similarly, the same attribute should be supported for Openness user during import as well so that you would be able to assign the possible proDiagFB for Array DB and instance DB of UDT.

XML structure of the exported Array DB

Use the following XML structure to export Array DB of UDT which gets assigned with ProdiagFB "Default_SupervisionFB":

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
<Engineering version="V17" />
<DocumentInfo>
<Created>2020-06-09T16:06:07.7850963Z</Created>
<ExportSetting>None</ExportSetting>
<InstalledProducts>
<Product>
<DisplayName>Totally Integrated Automation Portal</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<Product>
<DisplayName>Feature Cycle 1 TIA Portal</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<Product>
<DisplayName>Feature Cycle 3 TIA Portal</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<Product>
<DisplayName>STEP 7 Professional</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>STEP 7 Safety</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
</InstalledProducts>
</DocumentInfo>
<SW.Blocks.ArrayDB ID="0">
<AttributeList>
<Interface><Sections>
<Section Name="None">
<Member Name="Data_block_6" Datatype="Array[0..1] of &quot;Udt_With_Supervision&quot;">
<Comment>
<MultiLanguageText Lang="en-US">comment of Data_block_6</MultiLanguageText>
</Comment>
<Sections>
<Section Name="None">
<Member Name="Element_1" Datatype="Bool">
<Subelement Path="0">
<Comment>
<MultiLanguageText Lang="en-US">comment of Element_1</MultiLanguageText>
</Comment>
</Subelement>
<Subelement Path="1">
<Comment>
<MultiLanguageText Lang="en-US">comment of Element_1</MultiLanguageText>
</Comment>
</Subelement>
</Member>
</Section>
</Sections>
<Subelement Path="0">
<Comment>
```

```
<MultiLanguageText Lang="en-US">comment of Data_block_6[0]</MultiLanguageText>
</Comment>
<AssignedProDiagFB>Default_SupervisionFB</AssignedProDiagFB>
</Subelement>
<Subelement Path="1">
<Comment>
<MultiLanguageText Lang="en-US">comment of Data_block_6[1]</MultiLanguageText>
</Comment>
<AssignedProDiagFB>Default_SupervisionFB</AssignedProDiagFB>
</Subelement>
</Member>
</Section>
</Sections></Interface>
<Name>Data_block_6</Name>
<Number>17</Number>
<ProgrammingLanguage>DB</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="3" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="4" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.ArrayDB>
</Document>
```

Note

It is valid to have assignedProDiagFB only for Array DB of UDT, it is not possible to export any invalid data so you don't get any exception during export.

When you try to import the "AssignedProDiagFB" tag for any other ArrayDB which is not of UDT then you will encounter the not supported exception and this is not supported for System defined Datatype as well though they have bool variable or not.

Exported XML structure of Instance DB

Use the following XML structure to export instance DB of UDT which gets assigned with ProdiagFB "Default_SupervisionFB":

```
...
<SW.Blocks.InstanceDB ID="0">
<AttributeList>
<AssignedProDiagFB>Default_SupervisionDB</AssignedProDiagFB>
<InstanceOfName>User_data_type_4</InstanceOfName>
<InstanceOfType>UDT</InstanceOfType>
<Interface>
<Sections>
<Section Name="Static">
<Member Name="Element_1" Datatype="Bool" />
</Section>
</Sections>
</Interface>
<MemoryLayout>Optimized</MemoryLayout>
<Name>Data_block_9</Name>
<Number>19</Number>
<ProgrammingLanguage>DB</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="3" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="4" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.InstanceDB>
</Document>
...
```

Note

The appropriate "Not Supported attribute" exception should be thrown to you, If you try to set the assignedProDiagFB to instance DB.

See also

Opening a project (Page 128)

6.4.3 Technology objects

6.4.3.1 Overview of technology objects and versions

Technology objects

The following table shows the available technology objects for import and export.

CPU	Technology	Technology object	Version of technology object	CPU FW
S7-1200	Motion Control	TO_PositioningAxis	≥ V7.0	≥ V4.4
		TO_CommandTable		
	PID Control	PID_Compact	≥ V2.3	≥ V4.2
		PID_3Step	V2.3	
		PID_Temp	V1.1	

CPU	Technology	Technology object	Version of technology object	CPU FW
S7-1500	Motion Control	TO_SpeedAxis	≥ V5.0	≥ V2.8
		TO_PositioningAxis		
		TO_ExternalEncoder		
		TO_SynchronousAxis		
		TO_OutputCam		
		TO_CamTrack		
		TO_MeasuringInput		
		TO_Cam (S7-1500T)		
		TO_Kinematics (S7-1500T)		
		TO_LeadingAxisProxy (S7-1500T)		
		TO_Cam_10k (S7-1500T)	≥ V6.0	≥ V2.9
		TO_Interpreter (S7-1500T)	≥ V8.0	≥ V3.1
		TO_InterpreterMapping (S7-1500T)		
		TO_InterpreterProgram (S7-1500T)		
	Counting and measurement	High_Speed_Counter	≥ V4.1	Any
		SSI_Absolute_Encoder	≥ V3.1	
	PID Control	PID_Compact	≥ V2.3	≥ V2.0
		PID_3Step	V2.3	
		PID_Temp	V1.1	
		CONT_C	V1.1	Any
		CONT_S		
		TCONT_CP		
		TCONT_S		
S7-300/400	PID Control	CONT_C	V1.1	Any
		CONT_S		
		TCONT_CP		
		TCONT_S		
		TUN_EC		
		TUN_ES		
		PID_CP	V2.0	
		PID_ES		
	EMC	AXIS_REF	V2.0	

6.4.3.2 XML structure of the technology object interface section

Basic principle

The data in the XML file from the import/export is structured with reference to an openness XML file format. Every import file must fulfill the structural conditions.

The export file includes all edited tags and constants of the interface section of an exported technology object data block.

The project data can contain more data than the import XML file, e.g. external references. These must be exported separately.

Only writeable values can be imported via TIA Portal Openness XML.

Depending on the TIA Portal Openness export settings, the export file includes a defined set of attributes and elements. Export files from higher versions of the TIA Portal are not compatible with lower versions of the TIA Portal and could not be imported in those. SimaticML files exported with the latest TechObject version are also supported for the newer ones.

Basic structure

The SimaticML format is nearly the same as the export/import format of user blocks.

For technology objects, the following elements are required inside the element SW.TechnologicalObject.TechnologicalInstandDB.

```
<SW.TechnologicalObjects.TechnologicalInstanceDB ID="0">
<AttributeList>
  <InstanceOfName>TO_SpeedAxis</InstanceOfName>
  <Interface>
    ...
  </Interface>
  <Name>SpeedAxis_1</Name>
  <Number>1</Number>
  <OfSystemLibElement>TO_SpeedAxis</OfSystemLibElement>
  <OfSystemLibVersion>8.0</OfSystemLibVersion>
  <ParameterData>
    ...
  </ParameterData>
  <ProgrammingLanguage>Motion_DB</ProgrammingLanguage>
</AttributeList>
<ObjectList>
    ...
</ObjectList>
</SW.TechnologicalObjects.TechnologicalInstanceDB>
```

- InstanceOfName
  The name of the instance, from which the technology object is derived.
- Interface
  - Sections elements represent the hierarchy structure of the technology object datatypes and their version.
  - Member elements represent all parameters mapped directly to the technology object data block tags.
  - The AttributeList includes all defined attributes of a member. Attributes, that are system defined or assigned by a default value are not listed in the XML structure. The member attributes <ReadOnly> and <Informative> are only written to the XMLexport file if their value is TRUE.
- Name
  The name of the technology object in TIA Portal project
- Number
  The number of the technology object data block in TIA Portal project
- OfSystemLibElement
  The type of the technology object
- OfSystemLibVersion
  The version of the technology object
- ParameterData
  - Parameter elements represent parameters, that are not mapped directly to technology object data block tags or parameters, that are read-only technology object data block tags but writeable in the PublicAPI.
  - AdditionalData elements represent informations that are not stored in technology object data block tags, par example connections to drives or other technology objects.
- ProgrammingLanguage
  The only valid content is Motion_DB.

Parameters mapped directly to technology object data block tags

The sequence of the following description of elements represents parameters of a speed axis technology object mapped directly to technology object data blocks.

```
<Section Name="Static">
    ...
    <Member Name="DynamicLimits" Datatype="TO_Struct_DynamicLimits" Version="6.0">
      <AttributeList>
        <BooleanAttribute Name="SetPoint" SystemDefined="true">true</BooleanAttribute>
      </AttributeList>
      <Sections>
        <Section Name="None">
          <Member Name="MaxAcceleration" Datatype="LReal">
            <StartValue>1000.0</StartValue>
          </Member>
          <Member Name="MaxDeceleration" Datatype="LReal">
            <StartValue>1000.0</StartValue>
          </Member>
          <Member Name="MaxJerk" Datatype="LReal">
            <StartValue>20000.0</StartValue>
          </Member>
        </Section>
      </Sections>
    </Member>
    ...
</Section>
```

Parameters not mapped directly to technology object data block tags

The sequence of the following description of elements represents parameters of a speed axis technology object not mapped directly to technology object data blocks.

```
<ParameterData>
    <Parameters xmlns="http://www.siemens.com/automation/Openness/SW/Parameters/v1">
        <Parameter Name="_Units.VelocityUnit" />
        <Parameter Name="_Units.TorqueUnit" />
        <Parameter Name="Actor.Type" />
        <Parameter Name="Actor.Interface.EnableTorqueData" />
        <Parameter Name="Actor.Interface.EnableDriveOutput" />
        <Parameter Name="Actor.Interface.DriveReadyInput" />
        <Parameter Name="_Actor.Interface.EnableDriveOutputAddress" />
        <Parameter Name="_Actor.Interface.DriveReadyInputAddress" />
        <Parameter Name="_Actor.Interface.Telegram" />
        <Parameter Name="_Actor.DataAdaptionOffline" />
        <AdditionalData xmlns="http://www.siemens.com/automation/Openness/SW/Motion/Axis/v1/">
            <Connection Interface="Actor" OutputTag="myDrive1" />
            <Connection Interface="Torque" />
        </AdditionalData>
    </Parameters>
</ParameterData>
```

Additional information

For more information concerning Interface element and attributes of SimaticML elements see XML structure of the block interface section (Page 1295).

6.4.3.3 Exporting technology objects

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- A project contains TO to be exported.
- PLC is not online.

Application

The TIA Portal Openness API supports the export of all technology objects listed in the table Overview of technology objects and versions (Page 1401) to an XML file. You can only export consistent technology objects. Hardware configuration or tag tables are not exported with the technology object, but must be exported separately. You check the consistency of a technology object using the IsConsistent attribute. This flag can be set by successfully compiling of the respective technology object. The generation of the corresponding export file indicates that the export is completed.

In the following cases an exception is thrown:

- You try to export an inconsistent technology object.
- You try to export a technology object or a version of a technology object that does not support the export.
- You try to export a technology object while you are in online mode.

Parameters of Motion Control technology objects

Most parameters of Motion Control technology objects are directly mapped to technology object data block tags. These parameters will be exported in XML element Member in the Interface section.

Parameters that do not map directly to technology object data block tags will be exported in XML element ParameterData.

Detailed description of parameters:

- Specific parameters for TO_PositioningAxis and TO_CommandTable (Page 1409),
- Specific parameters for Axis and Encoder technology objects (Page 1411),
- Specific parameters for LeadingAxisProxy technology objects (Page 1415)
- Specific parameters for measuring input technology objects (Page 1415)
- Specific parameters for output cam and cam track technology objects (Page 1416)
- Specific parameters for Cam technology objects (Page 1417)
- Specific parameters for Kinematics technology objects (Page 1421)

Program code

```
void Export(FileInfo path, ExportOptions options);
```

The path parameter describes the path and file name of the export file that should be created. The parameter (ExportOptions options) specifies options for the export (Page 1211).

Attributes	Description
ExportOptions.None	Exports only modified parameters.
ExportOptions.WithDefaults	This option exports the parameters that have their default value. The default value itself is not exported. The same behavior is defined also for the TO specific parts of the XML export. The respective default values that are exported are defined for each TO type.
ExportOptions.WithReadOnly	This option exports additional informative attributes and values. During import, read only values are not written back to the DB. The respective read only values that are exported are defined for each TO type.
ExportOptions.WithDefaults | ExportOptions.WithReadOnly	Combination of the two options above.

Modify the following program code to find and export a technology object to an XML file:

```
// Find a specific technology object by its name and export this
private static void ExportTechnologicalObject(FileInfo path, ExportOptions options,
PlcSoftware plcSoftware, String nameOfTO)
{
    // Find by name
    TechnologicalInstanceDBComposition technologicalObjects = plcSoftware.TechnologicalObje
ctGroup.TechnologicalObjects;
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Find(nameOfTO);
    // Export TO
    technologicalObject.Export(path, options);
}
```

Modify the following program code to export a technology object OutputCam to an XML file:

```
// Export OutputCam
private static void ExportOutputCam(FileInfo path, ExportOptions options,
TechnologicalInstanceDB parentTO, String nameOfOutputCam)
{
    //Retrieve service OutputCamMeasuringInputContainer
    OutputCamMeasuringInputContainer container =
    parentTO.GetService<OutputCamMeasuringInputContainer>();
    //Get access to TO_OutputCam container
    TechnologicalInstanceDBComposition outputcamCamtrackContainer = container.OutputCams;
    //Find technology object TO_OutputCam
    TechnologicalInstanceDB outputCam = outputcamCamtrackContainer.Find("nameOfOutputCam");
    // Export
    outputCam.Export(path, options);
}
```

6.4.3.4 Importing technology objects

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is not online.

Application

The TIA Portal Openness API supports the import of technology objects from an XML file.

Exceptions

If the import data contains parameters that are not defined for the respective TO type, an EngineeringTargetInvocationException is thrown.

If the value of a parameter has a wrong format and cannot be converted to the type of the parameter, an EngineeringTargetInvocationException is thrown. The Import will be succeeded but an error will be produced on compile.

Open connection

In TIA Portal, open connections are created when a tag that is connected to a TO is deleted. The further behavior of open connections is identical to the behavior of open connections created by deleting a connected tag. If the partner TO is missing during the import, an open connection will be established in this case. This also applies to TO types that can not be used for certain connections.

Program code

```
IList<SW.TechnologicalObjects.TechnologicalInstanceDB>
Import(FileInfo path, ImportOptions options);
```

Modify the following program code to import one or several technology objects from an XML file. For more information about the ImportOptions see Importing configuration data (Page 1212).

```
// Import technology objects
private static void Import(FileInfo path, ImportOptions options, PlcSoftware plcSoftware)
{
    // Create technological instance
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;

    // Import TO
    technologicalObjects.Import(path, options);
}
```

6.4.3.5 S7-1200 Motion Control

Specific parameters for TO_PositioningAxis and TO_CommandTable

Application

The API supports the export and import of technology objects.

Parameters of TO_PositioningAxis and TO_CommandTable that are data block members are exported in the "Interface" section of the export file. Additional parameters are exported in the "ParameterData" section of the export file. The next chapter shows all additional parameters.

You can find a list of all available tags in SIMATIC STEP 7 S7-1200 Motion Control function manual on the internet (https://support.industry.siemens.com/cs/ww/en/view/109773400).

Note

The data type information is only exported for parameters that are part of the "Interface" section.

Additional parameters TO_PositioningAxis

The following additional parameters are not directly mapped to DB members and will be exported in section ParameterData depending on the ExportOption:

Parameter Name	Possible Values
_Actor.Interface.PTO	Pulse generator data: 0: Pulse_1 1: Pulse_2 2: Pulse_3 3: Pulse_4
_Actor.Interface.EnableDriveOutput	Tag name of the connected device address

_Actor.Interface.DriveReadyInput	Tag name of the connected device address
_Actor.Interface.DataConnection	0: Drive 1: DB
_Actor.Interface.DataBlock	See the functional view for possible values
_Actor.Interface.Analog	See the functional view for possible values
_Actor.Interface.ProfiDriveIn	Tag name of the connected device address
_Actor.Interface.ProfiDriveOut	Tag name of the connected device address
_Actor.DataAdaptionOffline	TRUE or FALSE
_Sensor[1].Interface.ProfiDriveIn	Tag name of the connected device address
_Sensor[1].Interface.ProfiDriveOut	Tag name of the connected device address
_ Sensor[1].Interface.EncoderConnection	4: HSC 7: Encoder on PROFINET/PROFIBUS
_Sensor[1].DataAdaptionOffline	TRUE or FALSE
_Sensor[1].Interface.DataConnection	0: Encoder 1: DB
_ Sensor[1].Interface.HSC	HSC number: 0: HSC_1 1: HSC_2 2: HSC_3 3: HSC_4
_ Sensor[1].Interface.DataBlock	See the functional view for possible values
_ Sensor[1].PassiveHoming.DigitalInput	Tag name of the connected device address
_ Sensor[1].ActiveHoming.DigitalInput	Tag name of the connected device address
_PositionLimits_HW.MinSwitch	Tag name of the connected device address
_PositionLimits_HW.MaxSwitch	Tag name of the connected device address

Additional parameters TO_CommandTable

The following additional parameters are not directly mapped to DB members:

Parameter name	Possible values
_WarningsEnable	TRUE or FALSE
_UseAxisParametersFrom	"Sample axis", Name of axis

External references

External references at the TO_PositioningAxis

The configuration window of TO_PositioningAxis displays information that is not stored in technology object data block tags. This external information is therefore not part of the exported xml file.

The tags used by the axis for assignment between symbolic names and addresses are stored in a tag table and must be exported and imported separately.

If the following parameters have not previously been set in the hardware configuration, they must be set by Writing parameters of technology object (Page 540).

In case of PTO-Axis:

An explicit set of _Actor.Interface.PTO to connect the technology object to the PTO-Output.

- _Actor.Interface.PTO_OutputA
- _Actor.Interface.PTO_OutputBEnable
- _Actor.Interface.PTO_OutputB
- _Actor.Interface.PTO_SignalType

Used digital inputs for homing

- _Sensor[1].ActiveHoming.DigitalInput
- _Sensor[1].PassiveHoming.DigitalInput

Used digital inputs for position limits

- _PositionLimits_HW.MaxSwitch
- _PositionLimits_HW.MinSwitch

In case of Analog/ProfiDrive:

An explicit set of following parameters to activate the PIP to OB-Servo:

- _Actor.Interface.ProfiDriveOutName
- _Actor.Interface.ProfiDriveInName
- _Sensor[1].Interface.ProfiDriveOutName
- _Sensor[1].Interface.ProfiDriveInName

In case of HSC is used for encoder connection:

An explicit set of _Sensor[1].Interface.HSC to connect the technology object to the HSC and activate the HSC in Hardware Configuration.

- _ Sensor[1].Interface.HSC_OperatingMode
- _ Sensor[1].Interface.HSC_InputA
- _ Sensor[1].Interface.HSC_InputB

6.4.3.6 S7-1500 Motion Control

Specific parameters for Axis and Encoder technology objects

Application

The API supports the import and export of technology objects. The following section describes the specific parameters.

Export TO_SpeedAxis, TO_PositioningAxis, TO_SynchronousAxis and TO_ExternalEncoder

Some parameters that map to read-only technology object data block tags are writeable in the PublicAPI. This parameters will be exported to ParameterData in the exported xml-file. The parameter name, the allowed values and default values are the same ones as for the corresponding data block tags.

The affected parameters are listed in the following tables:

Name	TO_SpeedAxis	TO_PositioningAxis / TO_SynchronousAxis	TO_ExternalEncoder
Actor.Type	X	X	-
Actor.Interface.EnableDriveOutput	X	X	-
Actor.Interface.DriveReadyInput	X	X	-
Actor.Interface.EnableTorqueData	X	X	-
VirtualAxis.Mode	X	X	-
Sensor[n].Existent[1)]	-	X	-
Sensor[n].Interface.Number[1)]	-	X	-
Sensor[n].Type[1)]	-	X	-
Sensor.Interface.Number	-	-	X
Sensor.Type	-	-	X
CrossPlcSynchronousOperation.Interface[n].EnableLeadingValueOutput[5)]	-	X	X

1) S7-1500 PLC: n=1; S7-1500T PLC: 1≤n≤4

4) ≥V5.0

5) V5.0 n=1; ≥V6.0 1≤n≤8

Some parameters do not map directly to technology object data block tags. This parameters will be exported to ParameterData in the exported xml-file. The parameter name, the allowed values and default values are the same ones as for the corresponding data block tags.

The affected parameters are listed in the following tables:

Name	**Default Value**	TO_SpeedAxis	TO_PositioningAxis / TO_SynchronousAxis	TO_ExternalEncoder
_Actor.DataAdaptionOffline	false	X	X	-
_Actor.Interface.Telegram	0	X	X	-
_Units.LengthUnit	1013 (mm)	-	X	X
_Units.VelocityUnit	1062 (mm/s)	1083 (1/mm)	X	X
_Units.TorqueUnit	1126 (Nm)	X	X	-
_Units.ForceUnit	1120 (N)	-	X	-
_Sensor[n].DataAdaptionOffline[1)]	false	-	X	-

_Sensor[n].Interface.Telegram[1)]	0	-	X	-
_Sensor.DataAdaptionOffline	false	-	-	X
_Sensor.Interface.Telegram	false	-	-	X
_Properties.MotionType	0	-	X	X
_Properties.UseHighResolution-PositionValues[4)]	false	-	X	X
_CrossPlcSynchronousOpera-tion.ActivateLocalLeadingValue-DelayTimeCalculation[4)]	true	-	X	X

Some parameters do not map directly to technology object data block tags. This parameters will be exported to ParameterData in the exported xml-file. In SimaticML, only the name of the connected tag is exported.

The affected parameters are listed in the following tables:

Name	TO_SpeedAxis	TO_PositioningAxis / TO_SynchronousAxis	TO_ExternalEn-coder
_Actor.Interface.EnableDriveOutputAddress	X	X	-
_Actor.Interface.DriveReadyInputAddress	X	X	-
_Sensor[n].ActiveHoming.DigitalInputAd-dress[1)]	-	X	-
_Sensor[n].PassiveHoming.DigitalInputAd-dress[1)]	-	X	-
_Sensor.ActiveHoming.DigitalInputAddress	-	-	X
_Sensor.PassiveHoming.DigitalInputAddress	-	-	X
_PositionLimits_HW.MinSwitchAddress	-	X	-
_PositionLimits_HW.MaxSwitchAddress	-	X	-
_CrossPlcOperation.Interface[n].Address-Out[5)]	-	X	X

XML element Connection

The XML element Connection describes the connections of a technology object. The names and values of the attributes listed below correspond to the parameter names and values of the particular AxisEncoderHardwareConnectionInterface.

- Required Attribute "Interface"
  - Possible values:

Value	TO_SpeedAxis	TO_PositioningAxis / TO_Synchro-nousAxis	TO_ExternalEncoder
Actor	X	X	-
Sensor	-	-	X
Sensor[n][1)]	-	X	-
Torque	X	X	-

1) S7-1500 PLC: n=1; S7-1500T PLC: 1≤n≤4

- Optional Attributes InputAddress and OutputAddress
  - Both attributes must either be present together or not be used at all
  - The possible values are bit addresses (as in API)
  - The attribute describes connections to DeviceItems and Channels
- Optional Attribute ConnectOption
  - May only be used together with InputAddress and OutputAddress
  - The attribute corresponds to method parameter of same name at AxisEncoderHardwareConnectionInterface.Connect
  - The attribute has the possible values Default and AllowAllModules
- Optional Attribute SensorIndexInActorTelegram
  - The attribute is used for connections to sensor part in actor telegram
  - The rules are the same as defined for the according API attribute
- Optional Attribute PathToDBMember
  - The attribute has the same possible values as for parameter of corresponding method
  - Describes a connection to a DB member
- Optional Attribute OutputTag
  - The value is name of connected PlcTag
  - The attribute describes a connection to an analog tag

Connect TO_SynchronousAxis with leading values

You connect TO_SynchronousAxis with leading values via the service SynchronousAxisMasterValues, see Connecting synchronous axis with leading values (Page 571).

Specific parameters for LeadingAxisProxy technology objects

Application

The API supports the import and export of technology objects. The following section describes the specific parameters.

Export TO_LeadingAxisProxy

The parameter _Interface.AddressIn does not map directly to technology object data block tags. These parameters will be exported to ParameterData in the exported xml-file. In SimaticML, only the name of the connected tag is exported.

Here the same rules apply as described in Specific parameters for Axis and Encoder technology objects (Page 1411).

Specific parameters for measuring input technology objects

Application

The API supports the import and export of technology objects. The following section describes the specific parameters.

Export TO_MeasuringInput

Some parameters that map to read-only technology object data block tags are writeable in the PublicAPI. These parameters will be exported to ParameterData in the exported xml-file.

Some parameters do not map directly to technology object data block tags. This parameters will be exported to ParameterData in the exported xml-file.

The parameter name, the allowed values and default values are the same ones as for the corresponding data block tags.

The attribute ParameterData contains the following parameters for TO_MeasuringInput:

- Parameter.MeasuringInputType
- _ListenToMeasuringInput

The parameter "_AssociatedObject" does not map directly to technology object data block tag. It will not be exported to ParameterData in the exported xml-file.

XML element "Connection"

The XML element Connection describes the connections of a technology object. The names and values of the attributes listed below correspond to the parameter names and values of the particular MeasuringInputHardwareConnectionProvider.

Value	Description
Interface	Single possible value: MeasuringInput
InputAddress	• The value is bit address • It describes a connection to a DeviceItem or a channel

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

Specific parameters for output cam and cam track technology objects

Application

The API supports the import and export of technology objects. The following section describes the specific parameters.

Export TO_OutputCam and TO_CamTrack

Some parameters that map to read-only technology object data block tags are writeable in the PublicAPI. These parameters will be exported to ParameterData in the exported xml-file. The parameter name, the allowed values and default values are the same ones as for the corresponding data block tags.

For TO_OutputCam the attribute ParameterData contains the parameter Interface.LogicOperation.

TO_CamTrack does not have any elements in the attribute ParameterData.

The parameter "_AssociatedObject" does not map directly to technology object data block tag. It will not be exported to ParameterData in the exported xml-file.

XML element "Connection"

The XML element Connection describes the connections of a technology object. The names and values of the attributes listed below correspond to the parameter names and values of the particular OutputCamHardwareConnectionProvider.

Value	Description	TO_OutputCam	TO_CamTrack
Interface	Single possible value: OutputCam	X	-

OutputAddress	• The value is bit address • It describes a connection to a channel	X	X
OutputTag	• The value is name of connected PlcTag • It describes a connection to an analog tag	X	X

Specific parameters for Cam technology objects

Application

The API supports the import and export of technology objects. The following section describes the specific parameters.

Export TO_Cam or TO_CAM_10k

The cam profile of a cam technology object of type TO_Cam or of type TO_CAM_10k will be exported to the element ParameterData.

The sequence of the following description of elements represents the structure of ParameterData.

```
<ParameterData>
    <Parameters xmlns="http://www.siemens.com/automation/Openness/SW/Parameters/v1">
        <ProfileData xmlns="http://www.siemens.com/automation/Openness/SW/Motion/Cam/v1">
            <GeneralConfiguration StandardContinuity="Acceleration"
                                  StandardOptimizationGoal="Velocity"
                                  InterpolationMode="CubicSpline"
                                  BoundaryConditions="FirstDerivative">
                <DesignLeadingRange Start="0" End="360" />
                <DesignFollowingRange Start="-1" End="1" />
            </GeneralConfiguration>
            <Elements>
                <Point X="0" Y="0" />
                <Line StartX="0" EndX="45" StartY="0" EndY="1" />
                <Line StartX="45" EndX="90" StartY="1" EndY="1" />
                <Point X="120" Y="0" />
            </Elements>
        </ProfileData>
    </Parameters>
</ParameterData>
```

XML structure of ParameterData

XML element	Description
ProfileData	The XML element ProfileData is a single child element of Parameters inside the EOM attribute ParameterData. Top level extension element, contains the XML element GeneralConfiguration followed by the XML element Elements.
GeneralConfiguration	The XML element GeneralConfiguration describes the general configuration that is valid for the complete cam profile. Additionally, the following optional attributes are used: • Attribute StandardContinuity Possible values are Position, Velocity, Acceleration (default value) and Jerk • Attribute StandardOptimizationGoal Possible values are None (default value), Velocity, Acceleration, Jerk and DynamicMoment • Attribute InterpolationMode Possible values are Linear, CubicSpline (default value) and BezierSpline • Attribute BoundaryConditions Possible values are NoConstraint (default value) and FirstDerivative
DesignLeadingRange	The XML element describes the leading value range of the curve definition. The following two optional attributes with datatype xsd:float are used: • Attribute Start Default value is 0 • Attribute End Default value is 360 The value of Start must be less than the value of End.
DesignFollowingRange	The XML element describes the following value range of the curve definition. • Attribute Start Default value is -1 • Attribute End Default value is 1

XML element	Description
Elements	The XML element contains the list of cam profile elements. Possible XML elements are: • Point • Line • Polynomial • VDITransition • Sine • InverseSine • PointGroup Except of the XML element Point, all XML Elements uses the required XML attributes StartX and EndX of type xsd:float. These attributes contain the X coordinate at the start respectively the end of the segment.
Point	The XML element describes a single point. The following attributes are used, all of them have the datatype xsd:float and default value 0: • Required Attribute X • Required Attribute Y • Optional Attribute Velocity • Optional Attribute Acceleration • Optional Attribute Jerk
Line	The XML element describes a Line segment. The following attributes are used, all of them have the datatype xsd:float and default value 0: • Optional Attribute StartY Contains the Y coordinate at the start of the line • Optional Attribute EndY Contains the Y coordinate at the end of the line • Optional Attribute Gradient Contains the gradient of the line Two of these three optional attributes must be present in the export.XML
Polynomial	The XML element describes a Polynomial segment. The XML element Polynomial has the sub elements TrigonometricValues, Coefficients and Constraints.TrigonometricValues can be used optionally, but either Coefficients or Constraints need to be present.

XML element	Description
TrigonometricValues	The XML element TrigonometricValues can be optionally used as sub element of Polynomial. It has the following attributes with datatype xsd:float: • Optional Attribute Amplitude Default value is 1 • Optional Attribute StartPhase Default value is 0 • Optional Attribute EndPhase Default value is 6.2831853071795862 • Optional Attribute Frequency Default value is 1 • Optional Attribute PeriodLength Default value is 1 Two of the attributes StartPhase, EndPhase, Frequency and PeriodLength must be used. Additionally, at least StartPhase or EndPhase needs to be present.
Coefficients	The XML element Coefficients can be optionally used as sub element of Polynomial.It has the optional attributes C0, C1, C2, C3, C4, C5 and C6. Each of them has the datatype xsd:float and the default value 0.
Constraints	This XML element can be optionally used as sub element of Polynomial. It has the following attributes with datatype xsd:float and default value 0: Optional Attribute LeftValue Optional Attribute RightValue Optional Attribute LeftVelocity Optional Attribute RightVelocity Optional Attribute LeftAcceleration Optional Attribute RightAcceleration Optional Attribute LeftJerk Optional Attribute RightJerk Optional Attribute Lambda Additionally, it has the optional attribute LambdaMode which supports the values Relative and Absolute (default value).
VDITransition	The XML element describes a VDI transition segment. It has the optional attributes LeftContinuity and RightContinuity which can have the values AsProfile (default value), Position, Velocity, Acceleration and Jerk. Additionally, the optional attribute OptimizationGoal with the possible values AsProfile (default value), None, Velocity, Acceleration, Jerk and DynamicMoment can be used.

XML element	Description
InverseSine	The XML element describes an InverseSine segment. It has the following optional attributes: • InterpolationPointCount • MaxFollowingValueTolerance • MathStartX • MathEndX • Minimum • Maximum • Inversed
Sine	The XML element describes a Sine segment. The Sine element has the following attributes: • Amplitude • StartPhase • EndPhase • Frequency • PeriodLength • Inclination • StartOffset • EndOffset
PointGroup	The XML element describes a PointGroup segment that contains several Points. The PointGroup element has the following optional attributes: • ApproximationDataPoints • ApproximationTolerance • LeadingValueMode • FollowingValueMode • ApproximationMode

Specific parameters for Kinematics technology objects

Application

The API supports the import and export of technology objects. The following section describes the specific parameters.

Export TO_Kinematics

Some parameters that map to read-only technology object data block tags are writeable in the PublicAPI. These parameters will be exported to ParameterData in the exported xml-file. The parameter name, the allowed values and default values are the same ones as for the corresponding data block tags.

- Kinematics.TypeOfKinematics
- MotionQueue.MaxNumberOfCommands

Some parameters do not map directly to technology object data block tags. These parameters will be exported to ParameterData in the exported xml-file. The parameter name, the allowed values and default values are the same ones as for the corresponding data block tags.

The affected parameters are listed in the following tables:

Name	Default value
_Units.LengthUnit	1013 (mm)
_Units.LengthVelocityUnit	1062 (mm/s)
_Units.AngleUnit	1126 (Nm)
_Units.AngleVelocityUnit	1120 (N)
_Properties.UseHighResolutionPositionValues[6)]	false
_X_Minimum	0.0
_X_Maximum	0.0
_Y_Minimum	0.0
_Y_Maximum	0.0
_Z_Minimum	0.0
_Z_Maximum	0.0
_A3_Maximum	0.0

XML structure of Connection

The XML element Connection describes the connections of a technology object.

The names and values of the attributes listed below correspond to the parameter names and values of the particular AxisEncoderHardwareConnectionInterface.

XML element	Description
AdditionalData	The element contains up to six KinematicsAxis elements, followed by one ConveyorTrackingLeadingValues element.
KinematicsAxis	The element describes a connected kinematics axis. This XML element has the following attributes: • Required Attribute Index The value of the index must be unique. – <V7.0 Possible values are 1..4 The index values corresponds to kinematics axis A1 to A4. – As of V7.0 Possible values are 1..6 The index values corresponds to kinematics axis A1 to A6. Kinematics axes A5 and A6 can only be used with "S7-1500T Motion Control KinPlus". • Required Attribute Ref Denotes the name of the connected axis technology object. This reference needs to be unique. • Required attribute Type Contains the type of the connected axis. The version of the connected axis is not nessassary.

XML element	Description
ConveyorTrackingLeadingValues	The element contains elements for SetPointCoupling followed by elements for ActualValueCoupling, DelayedCoupling.
SetPointCoupling	The element describes a connected leading value TO that is coupled via setpoint values. It has the following attributes: • Required Attribute Ref Denotes the name of the connected technology object which provides the leading value for conveyor tracking. • Required attribute Type Contains the associated TO type (version independent)
ActualValueCoupling	The element describes a connected leading value TO that is coupled via actual values. The same attributes as at SetPointCoupling elements are used.
DelayedCoupling	The element describes a connected leading value TO that is coupled via actual values. The same attributes as at SetPointCoupling elements are used.

Specific parameters for Interpreter technology objects

Application

The API supports the import and export of technology objects. The following section describes the specific parameters.

Export TO_Interpreter

The parameter Parameter.MaxNumberOfCommands maps to the read-only technology object data block tag that is writeable in the PublicAPI.

The parameter Parameter.MaxNumberOfCommands will be exported to ParameterData in the exported xml-file. The parameter name, the allowed values and default values are the same ones as for the corresponding data block tags.

Specific parameters for Interpreter program technology objects

Application

The API supports the import and export of technology objects.

The interpreter program technology object does not have any parameters.

Export TO_InterpreterProgram

The source code of the interpreter program technology object will be exported to the XML element SourceData.

The following sequence of elements represents the structure of ParameterData.

```
<ParameterData>
 <Parameters xmlns="http://www.siemens.com/automation/Openness/SW/Parameters/v1">
  <SourceData xmlns="http://www.siemens.com/automation/Openness/SW/Motion/InterpreterProgram/v1">
        PROGRAM main
          VAR
            SetPointInWCS: ARRAY[1..6] OF TO_Struct_Ipr_Frame;
          END_VAR
          powerOn( axis := Axis_001 );
          powerOn( axis := Axis_002 );
          home( Axis_001, mode := 7, sensor := 0 );
          home( Axis_002, mode := 7, sensor := 0 );
          posAbs( Axis_001, 180.0, dir := 3, v := 100.0 );
          ...
        END_PROGRAM
  </SourceData>
 </Parameters>
</ParameterData>
```

6.4.3.7 PID control

Specific attributes for PID_Compact

The API interface supports the export and import of technology objects. You can find a list of all available variables in "SIMATIC S7-1200/S7-1500 PID control" function manual on the internet (https://support.industry.siemens.com/cs/ww/en/view/108210036).

Export

For the PID_Compact technology object all parameters are directliy mapped to technology object data block tags. All parameters have the same data type and the default value as defined in the date block. The XML element `ParameterData` is empty.

Import

As of Version PID_Compact V3.0 all parameters are part of the import.

Up to PID_Compact V2.4 the following parameters for PID_Compact are not part of the import:

- PhysicalUnit
- PhysicalQuantity
- _Config.OutputSelect
- _Retain.CtrlParams.SetByUser

These parameters will have the default value after the import.

Note

Up to PID_Compact V2.4, remake the associated settings manually after an import of the export file. This can be done by using the function Writing parameters (Page 540).

Specific attributes for PID_3Step and PID_Temp

The API interface supports the export and import of technology objects. You can find a list of all available variables in "SIMATIC S7-1200/S7-1500 PID control" function manual on the internet (https://support.industry.siemens.com/cs/ww/en/view/108210036).

Export

For the PID_3Step and PID_Temp technology objects all parameters are directliy mapped to technology object data block tags. All parameters have the same data type and the default value as defined in the date block. The XML element `ParameterData` is empty.

Import

The following parameters for PID_3Step and PID_Temp are not part of the import:

- PhysicalUnit
- PhysicalQuantity

These parameters will have the default value after the import. If you want to modify them you have to use the function Writing parameters (Page 540).

Note

Remake the associated settings manually after an import of a PID_3Step or PID_Temp export file.

6.4.3.8 Counting

The API interface supports the export and import of technology objects. You can find a list of all available parameters in the product information "Parameters of technology objects in TIA Portal Openness" on the internet (https://support.industry.siemens.com/cs/ww/en/view/109744932).

For the TO types High_Speed_Counter and SSI_Absolute_Encoder all parameters correspond to TO DB members and have the data type and the default value as defined in the TO DB. The EOM attribute `ParameterData` is empty.

6.4.3.9 Easy Motion Control

The API interface supports the export and import of technology objects. You can find a list of all available parameters in the product information "Parameters of technology objects in TIA Portal Openness" on the internet (https://support.industry.siemens.com/cs/ww/en/view/109744932).

For the TO type AXIS_REF all parameters correspond to TO DB members and have the data type and the default value as defined in the TO DB. The EOM attribute `ParameterData` is empty.

6.4.4 Tag tables

6.4.4.1 Exporting PLC tag tables

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

One XML file is exported per PLC tag table.

The TIA Portal Openness API supports the export of all PLC tag tables from the system group and its subgroups.

Program code

Modify the following program code to export all PLC tag tables from the system group and its subgroups:

```
private static void ExportAllTagTables(PlcSoftware plcSoftware)
{
PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
// Export all tables in the system group
ExportTagTables(plcTagTableSystemGroup.TagTables);
// Export the tables in underlying user groups
foreach(PlcTagTableUserGroup userGroup in plcTagTableSystemGroup.Groups)
{
ExportUserGroupDeep(userGroup);
}
}
private static void ExportTagTables(PlcTagTableComposition tagTables)
{
foreach(PlcTagTable table in tagTables)
{
table.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", table.Name)),
ExportOptions.WithDefaults);
}
}
private static void ExportUserGroupDeep(PlcTagTableUserGroup group)
{
ExportTagTables(group.TagTables);
foreach(PlcTagTableUserGroup userGroup in group.Groups)
{
ExportUserGroupDeep(userGroup);
}
}
```

See also

Exporting configuration data (Page 1211)

6.4.4.2 Importing PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Program code

Modify the following program code to import PLC tag tables or a folder structure with PLC tag tables from an XML file into the system group or a user-defined group:

```
//Imports tag tables to the tag system group
private static void ImportTagTable(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTagTableComposition tagTables = plcTagTableSystemGroup.TagTables;
    tagTables.Import(new FileInfo(@"D:\Samples\myTagTable.xml"), ImportOptions.Override);
    // Or, to import into a subfolder:
    // plcTagTableSystemGroup.Groups.Find("SubGroup").TagTables.Import(new
FileInfo(@"D:\Samples\myTagTable.xml"), ImportOptions.Override);
}
```

See also

Notes on performance of TIA Portal Openness (Page 128)

6.4.4.3 Exporting an individual tag or constant from a PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

The API interface supports the export of a tag or constant from a PLC tag table to an XML file. Make sure that the tag table names used conform to the file naming conventions of your file system.

You can do an export with export settings ExportOptions.None. The following will not be exported, when they are set to the described value:

- Tag: ExternalAccessible (default value is true)
- Tag: ExternableWritable (default value is true)
- Tag: ExternalVisible (default value is true)
- Tag: LocalAddress (default value is empty)
- Constant: Value (default value is empty)
- Tag and Constant: DataTypeName (default value is empty string)
- Tag: IsSafety (true, if safety relevant, false, if not)

The comment of a tag or constant is only exported if at least one language is set for the comment. If the comment is not set for all project languages, this comment is only exported for the set project languages.

Note

PLC system constants

PLC system constants are excluded from export and import.

Program code

Modify the following program code to export a specific tag or constant from a PLC tag table to an XML file:

```
//Exports a single tag or constant of a controller tag table
private static void ExportTag(PlcSoftware plcSoftware, string tagName)
{
PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
PlcTag tag = plcTagTableSystemGroup.TagTables[0].Tags.Find(tagName);
if(tag != null)
{
tag.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", tag.Name)),
ExportOptions.WithDefaults);
}
}
private static void ExportUserConstant(PlcSoftware plcSoftware, string userConstantName)
{
PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
PlcUserConstant plcConstant =
plcTagTableSystemGroup.TagTables[0].UserConstants.Find(userConstantName);
if(plcConstant != null)
{
plcConstant.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", plcConstant.Name)),
ExportOptions.WithDefaults);
}
}
```

See also

Exporting configuration data (Page 1211)

Notes on performance of TIA Portal Openness (Page 128)

6.4.4.4 Importing an individual tag or constant into a PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

You can import either tags or constants in an import call.

Note

Constants can oly beimported as user constants.

Program code

Modify the following program code to import tag groups or individual tags and constants from an XML file:

```
//Imports tags into a plc tag table
private static void ImportTag(PlcSoftware plcSoftware, string tagtableName)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable == null) return;

    tagTable.Tags.Import(new FileInfo(@"D:\Samples\myTags.xml"), ImportOptions.Override);
}

//Imports constants into a plc tag table
private static void ImportConstant(PlcSoftware plcSoftware, string tagtableName)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable == null) return;

    tagTable.UserConstants.Import(new FileInfo(@"D:\Samples\myConstants.xml"),
ImportOptions.Override);
}
```

See also

Exporting configuration data (Page 1211)

Notes on performance of TIA Portal Openness (Page 128)

6.5 Importing/exporting hardware data

6.5.1 AML file format

Introduction

AutomationML is a neutral data format based on XML for the storage and exchange of plant engineering information, which is provided as open standard. Goal of AutomationML is to interconnect the heterogeneous tool landscape of modern engineering tools in their different disciplines, e.g. mechanical plant engineering, electrical design, HMI, PLC, robot control.

The class model used for the export and import of CAx data is based on the following AML standards:

- Whitepaper AutomationML Part 1 – AutomationML Architecture, November 2018
- Whitepaper AutomationML Part 2 – AutomationML Role Libraries, October 2014
- Whitepaper AutomationML – AutomationML Communication, September 2014
- Whitepaper AutomationML– AutomationML and eCl@ss Integration, November 2021
- Best Practice Recommendation Multilingual expressions in AutomationML, March 2017
- Best Practice Recommendation Modelling of Reference Designations, September 2017

Schema

The AutomationML data exchange model is described by the CAEX schema Version 2.15.

6.5.2 Pruned AML

Introduction

Pruning is the act of optimizing the content by removing certain things which are not necessarily to be provided. In case of external tools like EPLAN , the auto created sub module information within a hardware configuration has no significance with respect to EPLAN. Hence, these tools generate an AML file by removing the auto created sub module information from the hardware configuration. This file is called as pruned AML.

Generation of pruned AML

Generation of a Pruned AML is based on the following rules in order.

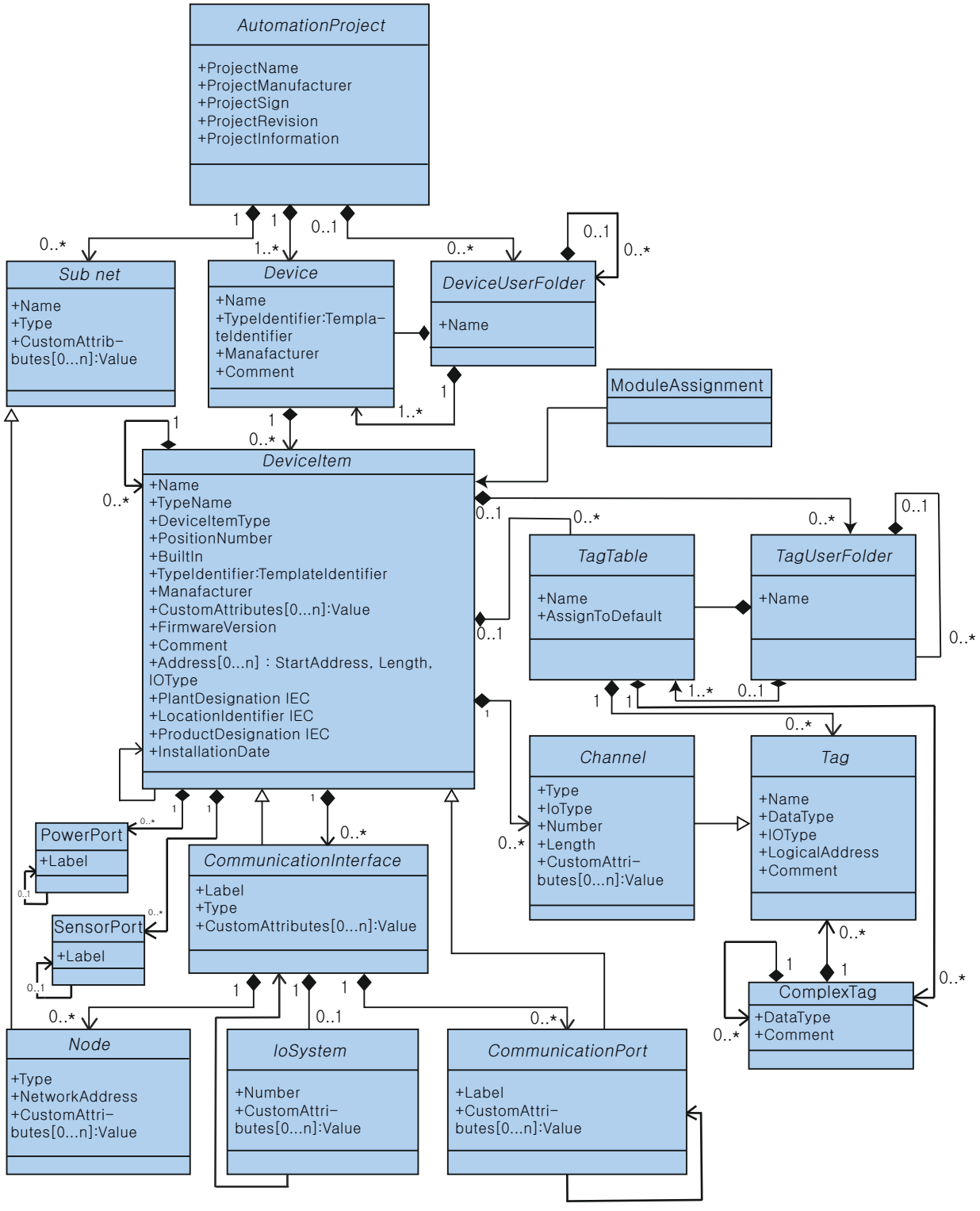
1. If a device item is pluggable, it shall not be pruned.
2. If a device item is of type "interface" or "port", it shall not be pruned.
3. If a device item is built-in and is at rack level, it shall not be pruned
4. AddressObjects of type "diagnosis" are not relevant for the prune algorithm.

5. Address objects linked with the auto created sub modules shall be provided under the immediate parent (which shall be a non-auto created sub module).
6. Address objects shall be included in the same sequence as returned by TIA Portal Openness.

6.5.3 Overview of the objects and parameters of the CAx import/export

Export/Import objects and attributes

The following figure shows the exportable objects with their attributes and dependencies of the CAx Import/Export.


AutomationProject
+ProjectName
+ProjectManufacturer
+ProjectSign
+ProjectRevision
+ProjectInformation
Sub net
+Name
+Type
+CustomAttrib-butes[0...n]:Value
Device
+Name
+TypeIdentifier:Templa-teIdentifier
+Manafacturer
+Comment
DeviceUserFolder
+Name
ModuleAssignment
DeviceItem
+Name
+TypeName
+DeviceItemType
+PositionNumber
+BuiltIn
+TypeIdentifier:TemplateIdentifier
+Manafacturer
+CustomAttributes[0...n]:Value
+FirmwareVersion
+Comment
+Address[0...n] : StartAddress, Length, IOType
+PlantDesignation IEC
+LocationIdentifier IEC
+ProductDesignation IEC
+InstallationDate
TagTable
+Name
+AssignToDefault
TagUserFolder
+Name
Channel
+Type
+IoType
+Number
+Length
+CustomAttri-butes[0...n]:Value
Tag
+Name
+DataType
+IOType
+LogicalAddress
+Comment
ComplexTag
+DataType
+Comment
PowerPort
+Label
SensorPort
+Label
CommunicationInterface
+Label
+Type
+CustomAttributes[0...n]:Value
Node
+Type
+NetworkAddress
+CustomAttri-butes[0...n]:Value
IoSystem
+Number
+CustomAttri-butes[0...n]:Value
CommunicationPort
+Label
+CustomAttri-butes[0...n]:Value

6.5.4 Structure of the CAx data for importing/exporting

Basic structure of an export file

The export file is generated in an AML format. The AML file starts with a document information.

The export file comprises the following two sections:

- Additional information
  The section includes information about the writer tool, referenced document versions (documents which defines the content of the AML file) etc.

Below XML depicts the AML file with latest AR APC recommendation corresponding to TIA Portal V19 exported file information which is showing <AddtionalInformation> section.

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" FileName="Project82.aml"
SchemaVersion="2.15" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
<WriterName>Totally Integrated Automation Portal</WriterName>
<WriterID>1d4fcebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
<WriterVendor>Siemens AG</WriterVendor>
<WriterVendorURL>www.siemens.com</WriterVendorURL>
<WriterVersion>19</WriterVersion>
<WriterRelease>1900.0000.0.0</WriterRelease>
<LastWritingDateTime>2023-07-19T04:21:32.9174411Z</LastWritingDateTime>
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.4.0" />
</AdditionalInformation>
...
...
</CAEXFile>
```

Note

CAx shall perform export and import of AR APC version in AML file in accordance to the installed version of TIA Portal.

- Instance hierarchy
  This section contains the hierarchical sequence of the exported internal elements.

```
...
  <InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="CAx_asterisk_AML_03_V14">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="544f3a69-5f65-45ba-ac2f-1448db9493fd" Name="PN/IE_1">
      ...
    </InternalElement>

    <InternalElement ID="12116ac0-94b7-49d2-888d-7d39bbc0caf5" Name="S71500/ET200MP station_1">
      ...
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
    <InternalLink Name="Link To Port_1" RefPartnerSideA=
      "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" RefPartnerSideB=
      "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" />
    <InternalLink Name="Link To Port_2" RefPartnerSideA=
      "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" RefPartnerSideB=
      "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" />
    .....
    <InternalLink Name="Link To IoSystem_3" RefPartnerSideA=
      "d8f6e006-3778-4a05-aab1-df844fe822fe:LogicalEndPoint_Interface" RefPartnerSideB=
      "2344b7af-329c-4215-92d1-6143b4627b56:LogicalEndPoint_IoSystem" />
  </InternalElement>
 </InstanceHierarchy>
</CAEXFile>
```

Internal elements

All objects inside the instance hierarchy of the AML file are `InternalElements`. The internal element `AutomationProject` contains all internal elements of all role classes. Every internal element supports a set of attributes.

The attribute `<TypeIdentifier>` identifies every object type of a hardware object that is creatable via TIA Portal Openness.

Note

Auto-created objects

Auto-created objects can only be created by other objects. They do not have properties or a type-identifier. They are included in the exported file, but you cannot trigger the export of a specific autocreated object.

At the end of the AML-element of an internal element, the following are defined:

- Role class
  The `SupportedRoleClass` element defines the object type of the internal element. The object type is defined in the role class library that maps the standard AML to the object model of TIA Portal Openness and TIA Portal.

```
...
<InternalElement ID="1d1a37ed-19d9-4a23-bc91-51f5a8e0244b" Name="Ungrouped devices">
  <InternalElement ID="ab193f5d-0375-4a6d-a576-a903e2b77cca" Name="ET 200SP station_1">
    ...
      <InternalElement ID="72d41729-90a7-4de3-9708-a8eeda6b1886" Name="IO device_1">
        ...
        <SupportedRoleClass RefRoleClassPath="
          AutomationProjectConfigurationRoleClassLib/DeviceItem" />
      </InternalElement>
      ...
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
  </InternalElement>
  ...
```

- Internal link
  The element `InternalLink` defines the communication partners of a connection.

```
  ...
  <InstanceHierarchy Name="APC Sample Instance Hierarchy">
    <InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="CAx_asterisk_AML_03_V14">
      <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
      <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
      <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
      <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
        ...
      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
      <InternalLink Name="Link To Port_1" RefPartnerSideA=
        "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" RefPartnerSideB=
        "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" />
      <InternalLink Name="Link To Port_2" RefPartnerSideA=
        "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" RefPartnerSideB=
        "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" />
      <InternalLink Name="Link To Port_3" RefPartnerSideA=
        "65307e3e-95fd-41ac-9982-e5e4ffc2fb15:CommunicationPortInterface" RefPartnerSideB=
        "58b1a3f2-f94b-48d1-ab5e-fbc4857cdfbc:CommunicationPortInterface" />
      <InternalLink Name="Link To Port_4" RefPartnerSideA=
        "58b1a3f2-f94b-48d1-ab5e-fbc4857cdfbc:CommunicationPortInterface" RefPartnerSideB=
        "65307e3e-95fd-41ac-9982-e5e4ffc2fb15:CommunicationPortInterface" />
      ...
    </InternalElement>
  </InstanceHierarchy>
</CAEXFile>
```

Attributes

Attributes are assigned to internal elements as follows:

```
...
<InternalElement ID="1d1a37ed-19d9-4a23-bc91-51f5a8e0244b" Name="Ungrouped devices">
  <InternalElement ID="ab193f5d-0375-4a6d-a576-a903e2b77cca" Name="ET 200SP station_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Device.ET200SP</Value>
    </Attribute>
    <InternalElement ID="7636c362-a7af-47bb-8a18-e6428a6d61ff" Name="Rack_0">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>Rack</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Rack.ET200SP</Value>
      </Attribute>
      ...
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
  <InternalLink Name="Link To Port_1" RefPartnerSideA=
      "5758e2ff-3974-41e9-8bcc-b61a23f1bb58:CommunicationPortInterface"
      RefPartnerSideB="46683602-5129-4504-a9d1-48e6421e2cf0:CommunicationPortInterface" />
  ...
</InternalElement>
```

Handling modes for attributes

The handling of attributes is defined for every attribute indiviually as follows:

- Ignored
  The attribute will be ignored during import and is not present in the export file.
- Mandatory
  The attribute has to be present in an import file and may not be deleted in the export file.
- Optional
  If this attribute is missing in the import file, the default value for the attribute is specified. This attribute is missing in an export file if it is not applicable for an object, e. g. not all modules have a FirmwareVersion.

- Export-only
  The attribute value is determined by the TIA Portal internally, e. g. the type name of a device item. If it is present in an import file, it will be ignored by the TIA Portal during import.
- Import-only
  The attribute can influence the import behavior. If the attribute is missing in an import file, the behavior will correlate to the standard value for the attribute.

See also

AML type identifiers (Page 1439)

6.5.5 AML type identifiers

Internal elements

The `TypeIdentifier` string consists of several parts:

- `<TypeIdentifierType>:<Identifier>`

The following values for `TypeIdentifierType` are possible:

- `OrderNumber` used to specify pyhsical modules
- `GSD` used to specify GSD/GSDML based devices
- `System` used to specify systems and generic devices

Type identifier type: OrderNumber

`OrderNumber` is the common type identifier for all modules present in the hardware catalog, excluding GSD. AML type identifier are not always equal to TIA Portal Openness type identifier. AML type identifier do not have a `FirmwareVersion` info. The information about firmware versions is handled in a separate AML attribute "`FirmwareVersion`".

The format for this `TypeIdentifierType` is as following:

- `<OrderNumber>`
  Example: `OrderNumber:3RK1 200-0CE00-0AA2`

Note

Wildcards in order numbers

There are a few modules in the hardware catalog which use "wildcard" characters in their order number to represent a certain cluster of real hardware, e. g. the different lengths of S7-300 racks.

In this case the specific `OrderNumber` and the "wildcard"-`OrderNumber` can both be used to create an instance of the hardware object. However, you cannot generically use wildcards at any position. Example: An S7-300 rack can be created in the following ways:

`OrderNumber:6ES7 390-1***0-0AA0`

or

`OrderNumber:6ES7 390-1AE80-0AA0`

Regard that you cannot use the following structure for instance:

`OrderNumber:6ES7 390-1AE80-0A*0`

The return value of reading the type identifier is always the order number from the hardware catalog.

Example: Reading `OrderNumber:6ES7 390-1AE80-0AA0`
returns `OrderNumber:6ES7 390-1***0-0AA0`

Type identifier type: GSD

The type identifier for GSD and GSDML based devices is `TypeIdentifier = GSD:<Identifier>`

The identifier is composed by the following elements

- `GsdName`: name of the GSD or GSDML in uppercase letters
- `GsdType`: One of the following:
  - D: Device
  - R: Rack
  - DAP: HeadModule
  - M: Module
  - SM: Submodule
- `GsdId`: ID of the GSD or GSDML

The following formats for the type identifier are supported of the CAx import/export:

- `GSD.<GsdName>/<GsdType>`
  Examples:
  `GSD:SIEM8139.GSD/DAP`
  `GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCP-20140313.XML/D`
- `<GsdName>/<GsdType>/<GsdId>`
  Examples:
  `GSD:SIEM8139.GSD/M/4`
  `GSD:GSDML-V2.31-SIEMENS-SINAMICS_G110M-20140704.XML/M/IDM_DRIVE_47`

Type identifier type: System

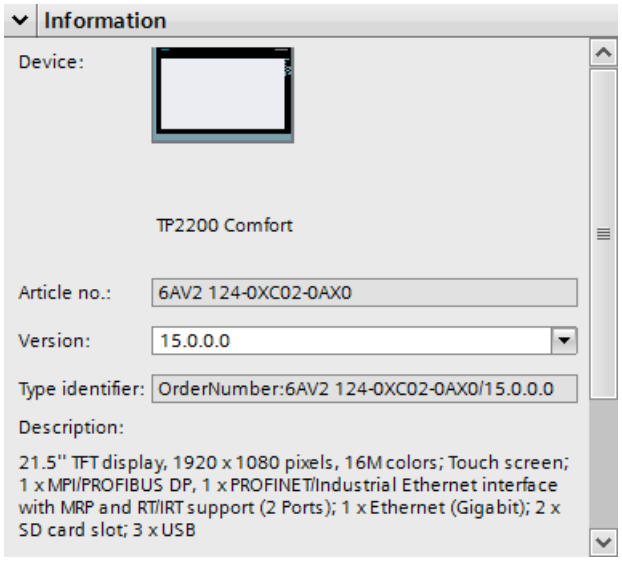
`System.` is the identifier for objects that cannot be determined by any other identifier. The formats for this `TypeIdentifierType` are as following:

- `<SystemTypeIdentifier>`
  Examples:
  `System:Device.S7300`
  `System:Subnet.Ethernet`
- `<SystemTypeIdentifier>/<AdditionalTypeIdentifier>`
  The `AdditionalTypeIdentifier` is necessary in case the `SystemTypeIdentifier` is not unique.
  The `SystemTypeIdentifier` has a prefix for certain object types:
  `Subnet.`
  `Device.`
  `Rack.`
  Example: `System:Rack.S71600/Large`
  A rack with an ordner number is identified via the `OrderNumber` identifier.

Displaying type identifiers in TIA Portal

If you need to know a type identifier you inquire it in TIA Portal as follows:

1. Enable the setting "Enable display of the type identifier for devices and modules" in "Options > Settings > Hardware configuration > Display of the type identifier".
2. Open the editor "Devices & networks".
3. Select a device in the Catalog.
   The type identifier is displayed in the viewlet "Information"



6.5.6 Export/Import of base unit Information via AML

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline

Application

In TIA Portal, you can export and import base unit information from and to TIA Portal so as to facililate the information exchange with other tools like EPLAN.

There are two types of base units supported during export and import in TIA Portal project:

- Single base unit
- Double base unit

Export of base unit

For example, a module configured with base unit information in TIA Portal, CAx shall export base unit information as a 'sub-module' under a module in the AML file.

The base unit submodule shall always be exported with:

- PositionNumber: 0
- DeviceItemType: Accessory
- BuiltIn: False
- TypeIdentifier: "The typeId of the Baseunit"
- ID: Always randomly generated GUID

Export of single base unit

The below example shows an AML file for DI module configured with a single base unit in TIA Portal

```
<InternalElement ID="6f76c890-5c5d-41c4-9ade-96543b0222ac" Name="DI 8x24VDC ST_1">
...
<InternalElement ID="69233c1f-7ef7-4999-8e84-691d0ff3a210" Name="BaseUnit">
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>Accessory</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6BP00-0DA0</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
...
```

Export of double base unit

The below example shows an AML file for two DI modules configured with double base unit in TIA Portal. For example, first module shall be configured with base unit having prefix 'IX300' and second shall be configured with same double base unit having prefix 'IX301'.

During export, only first module which is configured with base unit having IX300 suffix shall be exported with a base unit sub-module in the AML file. The second module, the one configured with IX301 shall not be injected with any Sub-module under it.

```
<InternalElement ID="6f76c890-5c5d-41c4-9ade-96543b0222ac" Name="DI 8x24VDC ST_1">
...
<InternalElement ID="3a1bee8a-12d0-4ec4-849c-333d45113d9c" Name="BaseUnit">
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>Accessory</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6BP60-0DA0</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
...
<InternalElement ID="5f843491-b053-4dc8-b879-9ac327ee2a7e" Name="DI 8x24VDC ST_2">
...
<InternalElement ID="55c30280-6f8a-4c37-9b2d-41bb90941258" Name="DI 8x24VDC ST_2">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value> </Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
...
```

Import of base unit

It shall be possible to import a module configured with a base unit sub-module in an AML file.

During import of base unit,

- GUID, PositionNumber, BuiltIn, and DeviceItem has no relevance, and hence it is not displayed in TIA Portal
- If any unintended module has base unit sub-module in AML file, Openness shallnot return "BaseUnit" attribute for it. Hence, CAx shall display a proper warning for it in the log file.
- CAx shall not verify for 'correctness' of base unit MLFB in AML file (except using it to identify whether it is single or double BU). CAx shall try to set the MLFB of base unit through Openness. In case of an error from Openness a proper error will be displayed.
- Import of previous version AML file with/without any BaseUnit information should be successful by importing information into TIA Portal.

Import of single base unit

During single base unit import, only the module carrying the base unit submodule in the AML file shall be imported with base unit information in TIA Portal.

Single BaseUnit shall be identified from the TypeIdentifier in AML file with a specific pattern as below:

OrderNumber:xxxx **193**-**6[B|U|T]**x**Y**x-xxxx where value of Y (11th Position) can be in the range of 0 to 5.

Import of double base unit:

During double base unit import, two modules(adjacent) shall be displayed with base unit information in TIA Portal. The first module configured with double base unit submodule in AML file shall be imported with a double base unit by appending suffix '|X300' to base unit MLFB. The second module, the one not having any base unit sub-module under it shall be imported with same double base unit by appending suffix '|X301' to base unit MLFB. Double BaseUnit shall be identified from the TypeIdentifier in AML file with a specific pattern as below:

OrderNumber:xxxx **193**-**6[B|U|T]**x**6**x-xxxx

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.7 Export/Import AML with extension rack connection

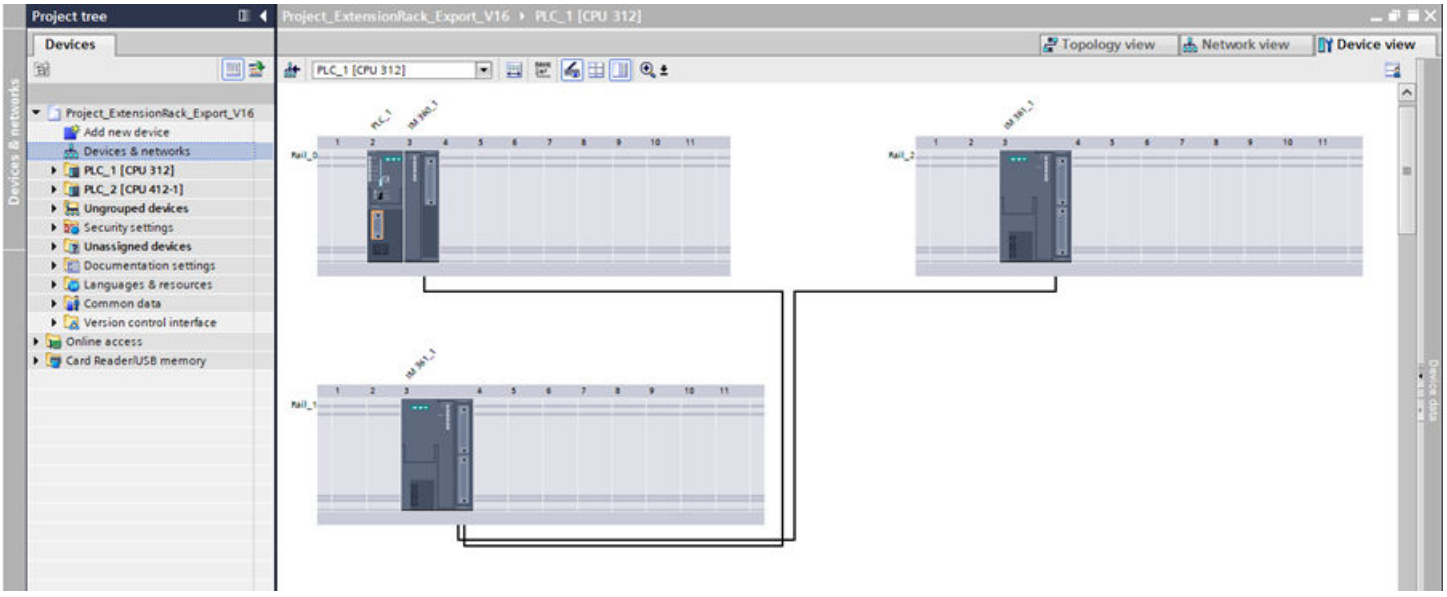
Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

In TIA Portal, you can export the devices with multiple racks having extension rack connection to an AML file and also import it back to get the same device configuration created back in the TIA Portal project.

Example: Device with multiple racks having extension rack connections



AML structure

In TIA Portal, extension rack connections between multiple racks are modeled under sender and receiver modules (both are DeviceItem objects) directly. However as per AR APC recommendation, these connections to be modeled as Port-to-Port connection under a CommunicationPort. A dummy CommunicationInterface having dummy CommunicationPort objects shall be added under IM modul to make inline with the recommendation.

The following sample shows a partial element structure of the exported AML file for above device configuration:

```
<InternalElement ID="1ddb8d5c-d6cc-42c9-b1d8-621219b139f6" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="f25e531a-1793-4896-ade5-a87bd98de06e" Name="IM 46x SenderPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="9a824a06-89b9-4ba8-bee0-83c89b1f5e53"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalElement ID="d841278f-558a-41ab-9f03-91eeb454dc6b" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="2e1ff3b2-8a3e-4d5b-a95c-3ed027287db9" Name="IM 46x ReceiverPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="98460c75-a05d-4c23-8f88-33878ccd79c5"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="7615a32e-09dc-4171-88ed-118026357bae" Name="IM 46x SenderPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X2</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
```

```
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="846093a9-6473-4946-acf4-95a7813924df"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalElement ID="f4eb31c6-41d4-4a6e-ac33-de9c054a8c74" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="c11d2227-91db-41ae-9d94-d822e3ab9c7a" Name="IM 46x ReceiverPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="a9b2cce1-7078-4347-b5f2-428da1ad5326"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalLink Name="Link To Port_1" RefPartnerSideA="f25e531a-1793-4896-ade5-
a87bd98de06e:CommunicationPortInterface" RefPartnerSideB="2e1ff3b2-8a3e-4d5b-
a95c-3ed027287db9:CommunicationPortInterface" />
<InternalLink Name="Link To Port_2"
RefPartnerSideA="7615a32e-09dc-4171-88ed-118026357bae:CommunicationPortInterface"
RefPartnerSideB="c11d2227-91db-41ae-9d94-d822e3ab9c7a:CommunicationPortInterface" />
```

Note

The extension rack interface shall be exported only when extension rack connections are present. Also, the number of dummy ports added under ExtensionRack dummy interface is based on ports participating in extension rack connection at module level. In above example, the "IM 460-0_1" module supports two ports (IM 46x SenderPort_1 and IM 46x SenderPort_2). However, only one port is configured with connection in TIA Portal. Hence, the exported AML file shall contain only one port under extension rack interface.

Extension rack connection

The XML representation of extension rack connections between multiple racks shall be done using format shown below.

ExternalInterface-

<ExternalInterface> internal element shall be added under <CommunicationPort> internal element which participates in the connection

```
<InternalElement ID="[IM Module Unique ID]" Name="[IM Module Name]">
...
<InternalElement ID="[Dummy Interface Unique ID]" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="[Dummy Port Unique ID]" Name="[IM Module Sender/Receiver Name]">
...
<ExternalInterface ID="[External Interface Unique ID]" Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="[Dummy Port Unique ID]" Name="[IM Module Sender/Receiver Name]">
...
<ExternalInterface ID="[External Interface Unique ID]" Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

Internal link

Extension racks connections are represented using <InternalLink> tags. The <InternalLink> tags shall be added under common parent of multiple racks (i.e., Device). Internal link name shall be unique across the common parent.

```
<InternalLink Name="Link To [Internal link Name]" RefPartnerSideA="[Communication Port
UniqueID]:[Communication Port External Interface Name]" RefPartnerSideB="[Communication
Port UniqueID]:[Communication Port External Interface Name]" />
```

6.5.8 Export/import AML file with GSD/GSDML custom attributes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

Custom attributes in general allow the exchange of data which are specific for a certain device or module and not covered by an AutomationML based specification. In TIA Portal, The custom attributes are supported for all types of modules like GSD, GSDML, non-GSD/GSDML based modules. Only
additional data of the modules can be exchanged by custom attributes but not additional data at ports, interfaces, nodes, etc.
Custom attributes are defined in the AR APC as an unordered list of name value pairs.

For information on non-GSD/GSDML custom attributes, see Export/Import AML file with non-GSD/GSDML custom attributes (Page 1458)

The AML structure is given here:"

```
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute  Name="AttributeName1" AttributeDataType="xs:string">
<Value>AttributeValue1</Value>
</Attribute>
<Attribute  Name="AttributeName2" AttributeDataType="xs:string">
<Value>AttributeValue2</Value>
</Attribute>
</Attribute>
```

Note

GSD/GSDML modules do not support channel configuration in TIA Portal. Hence, there shall be no support of export/import of channel custom attributes available via AML.

To allow a correct identification of a custom attribute in the TIA Portal, the name of the custom attribute must conform to the following definitions:

A custom attribute named consists of three parts <name>, <attribute_location>, <value_location>.

- **<name>** is a string containing only alphanumeric letters, numbers , "_" and ".". Usually the 'name' is the attribute name found in GSD/GSDML file. It is an optional part for attributes in parameter dataset otherwise mandatory.
- **<attribute_location>** defines the device item where the attribute is located. This is required to identify the location of the attribute if it is not located directly at the module. It has the form #<subslotnumber>#<subsubslotnumber>. That means, if the attribute is located at the module the <attribute_location> is empty, if it is located at a Sub Module it is "#<subslotnumber>" and in case of a location at a sub sub module the value is "#<subslotnumber>#<subsubslotnumber>".
- **<value location>** defines how to access an attribute at its device item. In case the value is part of a parameter data set, it is identified by the following quintuple:
  - **DatasetNumber**: The number of the data set at the device item.
  - **ByteOffset**: The position in bytes (starting with 0) where the value begins within the parameter data set.
  - **Length**: The overall length of the data set in bytes.
  - **BitOffset**: The position in bits (starting with 0) where the value begins within its start byte (see ByteOffset).
  - **BitLength:** The complete length of the value in bit

Overall example for custom attributes:

```
<Attribute name = "CustomAttribute">
<RefSemantic CorrespondingAttributePath="ListType"/>
<Attribute Name="IDTP_No_Unit_DIAG#0-1-0-2-7-1" AttributeDataType="xs:string">
<Value>1<Value>
</Attribute>
<Attribute Name="IDTP_LANG#0-1-1-2-0-4" AttributeDataType="xs:string">
<Value>89<Value>
<Attribute Name="IDTP_D_FREEZE#0-1-1-2-7-1" AttributeDataType="xs:string">
<Value>1<Value>
</Attribute>
</Attribute>
```

In some (rare) cases the attribute is explicitly modelled (not part of parameter data set) in the GSD/GSDML. Then the value is accessed via attribute name.

Note

On export TIA Portal always exports the complete data sets to allow partner applications full access of the contained data

```
<Attribute name = "CustomAttribute">
<RefSemantic CorrespondingAttributePath="ListType"/>
<Attribute Name="PrmData#0-1-0-2-0-16" AttributeDataType="xs:string">
<Value>128, 0<Value>
</Attribute>
</Attribute>
```

Export/import of an AML file with PRM Data custom attributes

With TIA Portal V17, it shall be possible to exchange Parameter Data of GSD/GSDML modules via CAx export and import. In AML file this Parameter Data shall be represented as Custom Attributes.

Export

Below snippet shows the format in which PrmData shall be expected in the AML file if it is found over a pluggable module.

```
...
<InternalElement ID="049f1260-7c97-458e-84bd-12682f943f19" Name="Slave_1">
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string"> <Value>GSD:SI018098.GSD/
DAP</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="PrmData-0-0-39-0-312" AttributeDataType="xs:string">
<Value>39,129,0,0,28,0,128,15,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,15,255,255,255,255, 255,255,255,255,255,255,255,255,255,255,255,255</
Value>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
```

Below snippet shows the format in which PrmData in case where it is available over a built-in submodule in tia portal, but in the AML file it is given at the pluggable parent.

```
<InternalElement ID="8fb83cae-ae30-45d2-9d4c-6db1154af02d" Name="IE-AS-i-LINK">
...
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="PrmData#0-130-0-4-0-32" AttributeDataType="xs:string">
<Value>0,0,131,0</Value>
</Attribute>
</Attribute>
<InternalElement ID="574a55bb-1209-4672-86bd-01ee9085eaf6" Name="DAP 1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
...
```

- In the above snippet custom attribute name "PrmData#0-130-0-4-0-32" has the following meaning:
  - PrmData#0: Name of the attribute and position number of the submodule containing it
  - 130: Dataset number
  - 0: Byte Offset
  - 4: Dataset length in bytes
  - 0: Bit Offset
  - 32: Bit length
- Custom attributes in AML file shall never appear on any built-in modules expect for built-in modules directly placed under the Rack.
- In case of a GSD/GSDML built-in module that has PrmData , the custom attribute shall be exported over the immediate pluggable parent or a built-in parent directly under rack, whichever is appropriate.
- In case of prunned files(files exported from E-plan) the built-in submodule which is the actual owner of the custom attribute shall not be present in the AML file.

Import

CAx shall support import of PrmData custom attributes over GSD/GSDML modules in the following cases:

- If a pluggable GSD/GSDML module has been provided with custom attributes.
- If the file has the built-in submodule prunned and the custom attributes are provided over the appropriate pluggable parent module.
- If the file is not pruned, it has the built-in submodules and the custom attributes are provided over the appropriate pluggable parent module

Custom attributes provided as individual attributes and not as full PrmDataset shall also be supported with CAx import. This kind of a format is expected from Eplan exported files.

Below snippet shows such a sample.

```
<InternalElement ID="1cf85c26-cc2a-4413-b91b-e4b55e183762" Name="Slave_1">
...
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="PSR-0-4-26-0-1" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="C4F-0-4-26-1-1" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="GPC-0-4-26-2-1" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="SFC-0-4-26-3-1" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="ACC-0-4-26-4-1" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="CMV3.1-0-4-26-5-1" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="MUPR-0-5-26-0-32" AttributeDataType="xs:string">
<Value>0,0,31,64</Value>
</Attribute>
<Attribute Name="TMR-0-9-26-0-32" AttributeDataType="xs:string">
<Value>7,255,225,236</Value>
</Attribute>
<Attribute Name="TSOLF-0-13-26-0-8" AttributeDataType="xs:string">
<Value>100</Value>
</Attribute>
<Attribute Name="PP-0-25-26-0-8" AttributeDataType="xs:string">
<Value>2</Value>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
...
```

In the example above all the individual custom attributes pertain to Data set number 0. But they try to alter the value of different byte(s) and bit(s).

- CAx import shall support both Integer and Hex formats(e.g 0x67) for the value of a custom attribute, but only one format at a time for any given attribute.

Export/import an AML file with Non PRM Data custom attributes

With TIA Portal V17 it shall be possible to exchange all attributes of GSD/GSDML modules that have read-write access via CAx export and import. In AML file these attributes shall be represented as Custom Attributes.

Export

CAx export shall ignore all such attributes that are read-only in TIA Portal, the custom attribute section in a TIA Portal exported AML file shall contain attributes that are writable.

Below snippet shows the format in which custom attributes shall be expected in the AML file if it is found over a pluggable module.

```
...
<InternalElement ID="806532c8-109a-42c6-82e9-84e8ba308aad" Name="cp1604">
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="Author" AttributeDataType="xs:string">
<Value>AuthorValue</Value>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

Below snippet shows the format in which custom attribute shall be exported in case where it is available over a built-in submodule in TIA Portal, but in the AML file it is given at the pluggable parent.

```
...
<InternalElement ID="806532c8-109a-42c6-82e9-84e8ba308aad" Name="cp1604">
...
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="Failsafe_FIODBNumber#0" AttributeDataType="xs:string">
<Value>0</Value>
</Attribute>
<Attribute Name="Failsafe_FParameterSignatureIndividualParameters#0"
AttributeDataType="xs:string">
<Value>5</Value>
</Attribute>
</Attribute>
<InternalElement ID="6b94abcd-3fe4-4800-9e18-ea7810d7afed" Name="PS_8Byte">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

In the snippet above #0 signifies the position number of the built-in child item to which the attribute really belongs to.

CAx export shall ignore all attributes that are already being exported as part of the applicable AR APC version recommendation.

CAx shall not export custom attributes over interfaces and ports and any submodule under them. The only exception being PrmData.

Import

CAx shall support import of custom attributes over GSD/GSDML modules in the following cases:

- If a pluggable GSD/GSDML module has been provided with custom attributes.
- If the file has the built-in submodule pruned and the custom attributes are provided over the appropriate pluggable parent module.
- If the file is not pruned, it has the built-in sub modules and the custom attributes are provided over the appropriate pluggable parent module.

CAx import shall ignore all attributes with appropriate warning that are already being imported as part of the applicable AR APC version recommendation.

Note

Import of custom attributes may be limited when the attributes have complex dependency hierarchies on other attributes. For example: Attribute A is depending on Attribute B and Attribute C. So, the import of A works only after importing B and C. This behavior can vary from module to module. In such cases, import might not succeed and shall result in skipping those attributes (with a notification to the user about skipping) and the user shall have to explicitly configure failed/skipped custom attributes in TIA Portal after AML import has been done.

See also

Export/Import AML file with non-GSD/GSDML custom attributes (Page 1458)

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.9 Export/Import AML file with non-GSD/GSDML custom attributes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can use the TIA Portal V18 to exchange all non-GSD/GSDML attributes of hardware (sub)modules and channels that have Openness read-write access via CAx export and import. In AML file, these attributes shall be represented as custom attributes.

Export/Import of an AML file with module custom attributes

Export

CAx export shall consider only attributes which are read-writable in TIA Portal Openness and same shall be exported into AML file. In TIA Portal, the attributes are distributed at different levels. For ex: Pluggable (sub)modules, builtin (sub)modules. Irrespective this different

hierarchy levels, AML file shall always contain custom attributes at pluggable (sub)module level when configuration is exported:

1. When a pluggable (sub)module contains attributes, the attributes shall remain at same level in the exported AML file.
2. When a built-in (sub)module contains attributes, the attributes shall be moved to its immediate pluggable parent in the exported AML file

Following is the identifying name format for custom attribute in the AML file:

- TypeIdentifier.AttributeName#PositionNumber where

Attribute	Description
TypeIdentifier	Normalized TypeIdentifier (including FW version) of a pluggable (sub)module which is formatted to replace special characters with '_'
AttributeName	The name of the attribute in Openness.
PositionNumber	This is applicable only for attributes of built-in (sub)modules which are exported under its pluggable parent module in AML file. This is the position number of built-in (sub)module.

Below snippet shows the format in which custom attributes shall be expected in the AML file if it is found over a pluggable module.

```
<InternalElement ID="298d6b6a-fe70-4edd-9230-39b0ae2238a5" Name="PLC_1">
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.Author" AttributeDataType="xs:string">
<Value>cvdfff</Value>
</Attribute>
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.CentralAlarmManagement"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.ClockMemoryByte" AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.CommunicationMode" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.ConfigurationControl"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
</InternalElement>
```

Below snippet shows the format in which custom attribute shall be exported in case where it is available over a built-in submodule in TIA Portal, but in the AML file it is given at the pluggable parent.

```
...
<InternalElement ID="298d6b6a-fe70-4edd-9230-39b0ae2238a5" Name="PLC_1">
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.DisplayDefaultLanguage#3"
AttributeDataType="xs:string">
<Value>0</Value>
</Attribute>
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.DisplayProtection#3"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.DisplayTimeToEnergySavingMode#3"
AttributeDataType="xs:string">
<Value>900</Value>
</Attribute>
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.DisplayTimeToStandbyMode#3"
AttributeDataType="xs:string">
<Value>1800</Value>
</Attribute>
</InternalElement>
```

In the snippet above #3 signifies the position number of the built-in child item to which the attribute really belongs to in TIA Portal.

Note

CAx export shall ignore all attributes that are already being exported as part of the applicable AR APC version recommendation.

Import

CAx shall support import of custom attributes over (sub)modules in the following cases:

- If a pluggable (sub)module has been provided with custom attributes.
- If the file has the built-in submodule pruned and the custom attributes are provided over the appropriate pluggable parent (sub)module.
- If the file is not pruned, it has the built-in sub modules and the custom attributes are provided over the appropriate pluggable parent (sub)module.

CAx import shall ignore all attributes with appropriate warning that are already being imported as part of the applicable AR APC recommendation.

Tolerant Import

During Import, TIA Portal is tolerant with respect to custom attribute names. The prefix "TypeIdentifier." shall be optional.

- CAx import accepts AML file with custom attributes having complete IdentifyingName (format: TypeIdentifier.AttributeName#PositionNumber)
- CAx import accepts AML file with custom attributes having partial IdentifyingName - only attribute name and suffixes (like #positionnumber) but no TypeIdentifier

Below snippet shows different name formats accepted during CAx import.

```
...
<InternalElement ID="298d6b6a-fe70-4edd-9230-39b0ae2238a5" Name="PLC_1">
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7516_3AN03_0AB0_V2_9.CentralAlarmManagement"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute Name="CentralAlarmManagement" AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
</InternalElement>
```

Export/Import of an AML file with channel custom attributes

Export

CAx export shall consider only attributes under channel which are read-writable in Openness and same shall be exported into AML file. In TIA Portal, the channels are distributed at different levels. For ex: Pluggable (sub)modules, builtin (sub)modules. And, the AML file shall always contain channels (along with the attributes) at same level when configuration is exported.

Following is the identifying name format for channel custom attribute in the AML file.

- <TypeIdentifier>.Channels.<Type><IoType>_<ChannelNumber>.<AttributeName>#<PositionNumber> where

Custom Attribute	Description
TypeIdentifier	Normalized TypeIdentifier (including FW version) of a pluggable (sub)module which is formatted to replace special characters with '_'.
AttributeName	The name of the attribute in Openness
PositionNumber	This is applicable only for attributes of channels where channels are modelled under built-in (sub)modules. This is the position number of built-in (sub)module.
Type	This is type of channel, Analog or Digital (shortly abbreviated as A or D)

Custom Attribute	Description
IoType	This is IO type of channel, Input or Output (shortly abbreviated as I or O)
ChannelNumber	This is sequential integer number given to each channel starting from 0 to N

Below snippet shows the format in which custom attributes shall be expected in the AML file if the channel is found over a pluggable module.

```
...
<InternalElement ID="928ac5b5-2625-4f40-a624-d731ed673522" Name="DO 8x24VDC/
0.5A_1"><Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>DO8 x 24VDC / 0.5A</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>5</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 322-8BF00-0AB0</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7322_8BF00_0AB0.Author" AttributeDataType="xs:string">
<Value>Z003NH6D</Value>
</Attribute>
</Attribute>
<ExternalInterface ID="143cf656-2a27-4673-9a6a-55bc570068f6" Name="Channel_DO_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Digital</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Output</Value>
</Attribute>
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7322_8BF00_0AB0.Channels.DO_0.DiagnosticsNoSupplyVoltage"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7322_8BF00_0AB0.Channels.DO_0.DiagnosticsShortCircuitToGround"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7322_8BF00_0AB0.Channels.DO_0.DiagnosticsShortCircuitToLplus"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7322_8BF00_0AB0.Channels.DO_0.DiagnosticsWireBreak"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7322_8BF00_0AB0.Channels.DO_0.SubstituteValue"
AttributeDataType="xs:string"><Value>False</Value>
</Attribute>
```

```
</Attribute>
```

Below snippet shows the format in which custom attribute shall be exported if it is found over Built-in submodule

```
...
<InternalElementID="5bda81fa-8c31-4e1a-b41a-f5100eb2ff2f"Name="AI5/AQ2_1">
<AttributeName="PositionNumber"AttributeDataType="xs:int">
<Value>8</Value>
</Attribute>
<AttributeName="BuiltIn"AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<AttributeName="Address">
<RefSemanticCorrespondingAttributePath="OrderedListType"/>
<AttributeName="1">
<AttributeName="StartAddress"AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<AttributeName="Length"AttributeDataType="xs:int">
<Value>80</Value>
</Attribute>
<AttributeName="IoType"AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
<AttributeName="2">
<AttributeName="StartAddress"AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<AttributeName="Length"AttributeDataType="xs:int">
<Value>32</Value>
</Attribute>
<AttributeName="IoType"AttributeDataType="xs:string">
<Value>Output</Value>
</Attribute>
</Attribute>
</Attribute>
<ExternalInterfaceID="76a8ea3f-1da6-4622-
b806-912e1da53980"Name="Channel_AI_0"RefBaseClassPath="AutomationProjectConfigurationInter
faceClassLib/Channel">
<AttributeName="Type"AttributeDataType="xs:string">
<Value>Analog</Value>
</Attribute>
<AttributeName="IoType"AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
<AttributeName="Number"AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<AttributeName="Length"AttributeDataType="xs:int">
<Value>16</Value>
</Attribute>
<AttributeName="CustomAttributes">
<RefSemanticCorrespondingAttributePath="ListType"/>
<AttributeName="6ES7511_1CK00_0AB0_V2_9.Channels.AI_0.HardwareInterruptLowLimit2Active#8"A
ttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<AttributeName="6ES7511_1CK00_0AB0_V2_9.Channels.AI_0.OperatingRange#8"AttributeDataType="
```

```
xs:string">
<Value>OperatingRange.PlusMinus10V</Value>
</Attribute>
<AttributeName="6ES7511_1CK00_0AB0_V2_9.Channels.AI_0.OperatingType#8"AttributeDataType="x
s:string">
<Value>OperatingType.Voltage</Value>
</Attribute>
<AttributeName="6ES7511_1CK00_0AB0_V2_9.Channels.AI_0.ParameterSettings#8"AttributeDataTyp
e="xs:string">
<Value>ParameterSettings.Manual</Value>
</Attribute>
<AttributeName="6ES7511_1CK00_0AB0_V2_9.Channels.AI_0.Smoothing#8"AttributeDataType="xs:st
ring">
<Value>Smoothing.None</Value>
</Attribute>
</Attribute>
</ExternalInterface>
```

Note

In the snippet above #8 signifies the position number of the built-in child item to which the channel really belongs to in TIA Portal. CAx export shall ignore all attributes that are already being exported as part of the applicable AR APC version recommendation.

Import

CAx shall support import of channel's custom attributes in the following cases:

- If the file is not pruned and channels with its custom attributes are under (sub)module.
- If the file has the built-in submodule pruned and the channels with its custom attributes are provided over the appropriate pluggable parent (sub)module.
- If the file has the built-in submodule pruned and channels are moved under pluggable parent and channel's custom attributes are moved under pluggable device item custom attributes section with proper channel info and target device info.

All custom attributes supporting set of enum values shall be exported with fully qualified enum text value but while importing both fully qualified enum text value as well as its equivalent integer value.

Below are snippet shows channel custom attributes.

```
<InternalElement ID="0d5e860b-6581-467a-be7a-fcfa37e0ee6b" Name="DI 32x24VDC HF_1">
...
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 521-1BL00-0AB0</Value>
</Attribute>
<InternalElement ID="198bf5e3-ac19-4465-8e3e-8c2faf6ab217" Name="DI 32x24VDC HF_1">
...
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="ff89bfc5-ee51-4525-a388-a9b02213515e" Name="Channel_DI_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
...
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.DiagnosticsNoSupplyVoltage#1"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.DiagnosticsWireBreak#1"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
...
</Attribute>
</ExternalInterface>
...
```

```
<InternalElement ID="0d5e860b-6581-467a-be7a-fcfa37e0ee6b" Name="DI 32x24VDC HF_1">
...
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 521-1BL00-0AB0</Value>
</Attribute>
<ExternalInterface ID="ff89bfc5-ee51-4525-a388-a9b02213515e" Name="Channel_DI_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
...
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.DiagnosticsNoSupplyVoltage#1"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.DiagnosticsWireBreak#1"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
...
</Attribute>
</ExternalInterface>
...
```

Tolerant Import

During Import, TIA Portal is tolerant wrt channel custom attribute names. The prefix "TypeIdentifier" or channel info are optional.

Channel custom attribute is available on the channel:

- CAx import accepts AML file with custom attributes having complete IdentifyingName (format: TypeIdentifier.Channels.TypeIoType_ChannelNumber.AttributeName#PositionNumber)
- CAx import accepts AML file with custom attributes having partial IdentifyingName - only attribute name and suffixes (like #positionnumber) but no TypeIdentifier and channel info.
- Below are the supported formats:
  - <TypeIdentifier>.Channels.<Type><IoType>_<ChannelNumber>.<AttributeName>#<PositionNumber>
  - <AttributeName>#<PositionNumber>

Below snippet shows different name formats accepted during CAx import.

```
...
<ExternalInterface ID="ff89bfc5-ee51-4525-a388-a9b02213515e" Name="Channel_DI_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
...
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.DiagnosticsNoSupplyVoltage#1"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute Name="Channels.DI_0.DiagnosticsWireBreak#1" AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
...
</Attribute>
</ExternalInterface>
...
```

Channel custom attribute is available on the pluggable device item:

- CAx import accepts AML file with custom attributes having complete IdentifyingName (format: TypeIdentifier.Channels.TypeIoType_ChannelNumber.AttributeName#PositionNumber)
- CAx import accepts AML file with custom attributes having partial IdentifyingName - only attribute name with channel info (Channels.<Type><IoType>_<ChannelNumber>) and suffixes (like #positionnumber) but no TypeIdentifier.
- Below are the supported formats:
  - <TypeIdentifier>.Channels.<Type><IoType>_<ChannelNumber>.<AttributeName>#<PositionNumber>
  - Channels.<Type><IoType>_<ChannelNumber>.<AttributeName>#<PositionNumber>

Below snippet shows different name formats accepted during CAx import.

```
...
<ExternalInterface ID="ff89bfc5-ee51-4525-a388-a9b02213515e" Name="Channel_DI_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
...
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.DiagnosticsNoSupplyVoltage#1"
AttributeDataType="xs:string
<Value>True</Value>
</Attribute>
<Attribute Name="Channels.DI_0.DiagnosticsWireBreak#1" AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
...
</Attribute>
</ExternalInterface>
...
```

Invalid Channel Custom Attribute

CAx shall not support import of channel's custom attributes in the following cases:

- If channel's custom attributes are distributed under both device item custom attribute section and channel's custom attributes section, in that case only the channel custom attributes which are under device item will be processed and remaining custom attributes which are under channels will be ignored with warning.

Below snippet shows channel custom attributes distributed under device item and channels.

```
<InternalElement ID="0d5e860b-6581-467a-be7a-fcfa37e0ee6b" Name="DI 32x24VDC HF_1">
..
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 521-1BL00-0AB0</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Author" AttributeDataType="xs:string">
<Value>z003tyvt</Value>
</Attribute>
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.EnableValueStatus" AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.StartupComparisonPresetToActualModule"
AttributeDataType="xs:string">
<Value>0</Value>
</Attribute>
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.DiagnosticsNoSupplyVoltage#1"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.DiagnosticsWireBreak#1"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
</Attribute>
...
<ExternalInterface ID="ff89bfc5-ee51-4525-a388-a9b02213515e" Name="Channel_DI_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
...
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute
Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.HardwareInterruptFallingEdgeActive#1"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute
Name="6ES7521_1BL00_0AB0_V2_2.Channels.DI_0.HardwareInterruptFallingEdgeEventName#1"
AttributeDataType="xs:string">
<Value>Falling edge0</Value>
</Attribute>
</ExternalInterface>
...
```

Note

Import of custom attributes may be limited when the attributes have complex dependency hierarchies on other attributes. For example: Attribute A is depending on Attribute B and Attribute C. So, the import of A works only after importing B and C. This behavior can vary from module to module. In such cases, import might not succeed and shall result in skipping those attributes (with a notification to the user about skipping) and the user shall have to explicitly configure failed/skipped custom attributes in TIA Portal after AML import has been done.

Export/Import AML file with network object specific custom attributes

CAx shall export/import read-write custom attributes on various network objects like interface, port, node, IO system and subnet.

NetworkInterface configuration

Following is the identifying name format for Interface custom attribute in the AML file.

- <TypeIdentifier>.Interface.<Label>.<AttributeName> where

Custom attribute	Description
TypeIdentifier	Normalized TypeIdentifier (including FW version) of a pluggable (sub)module which is formatted to replace special characters with '_'.
AttributeName	The name of the attribute in Openness

Below snippet shows the format in which custom attributes shall be expected in the AML file if the Interface is found over a pluggable module.

```
...
<InternalElement ID="d539b22a-9ff4-4cf7-b0f4-2f46c652f82a" Name="PROFINET interface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32768</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute
Name="6DL4158_3FH04_3XX0_V4_1_S74100.Interfaces.X1.DeviceReplacementWithoutExchangeableMed
ium" AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute
Name="6DL4158_3FH04_3XX0_V4_1_S74100.Interfaces.X1.DiagnosticsCommunicationError"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="6DL4158_3FH04_3XX0_V4_1_S74100.Interfaces.X1.DisplayUpdateInterval"
AttributeDataType="xs:string>
<Value>DisplayUpdateInterval.Value10Seconds</Value>
</Attribute>
<Attribute Name="6DL4158_3FH04_3XX0_V4_1_S74100.Interfaces.X1.IECV22LLDPMode"
AttributeDataType="xs:string"<Value>True</Value>
</Attribute>
<Attribute Name="6DL4158_3FH04_3XX0_V4_1_S74100.Interfaces.X1.KeepAlivesInterval"
AttributeDataType="xs:string"<Value>30</Value>
</Attribute>
<Attribute Name="6DL4158_3FH04_3XX0_V4_1_S74100.Interfaces.X1.PnSendClock"
AttributeDataType="xs:string"><Value>1000000</Value>
</Attribute>
<Attribute Name="6DL4158_3FH04_3XX0_V4_1_S74100.Interfaces.X1.TimeSynchronizationNtp"
AttributeDataType="xs:string
<Value>True</Value>
</Attribute>
```

Port configuration

Following is the identifying name format for Port custom attribute in the AML file.

- <TypeIdentifier>.Interfaces.<InterfaceLabel>.Ports.<PortLabel>.<AttributeName> where

Custom attribute	Description
TypeIdentifier	Normalized TypeIdentifier (including FW version) of a pluggable (sub)module which is formatted to replace special characters with '_'.
AttributeName	The name of the attribute in Openness

Below snippet shows the format in which custom attributes shall be expected in the AML file if the Port is found over Interface.

```
...
<InternalElement ID="fc31e87d-2162-43f6-ae4c-211fb5e21dec" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32769</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Ports.P1R.AlternativePartnerPorts"
AttributeDataType="xs:string"><Value>False</Value>
</Attribute>
<Attribute
Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Ports.P1R.EndOfDetectionOfAccessibleDevices"
AttributeDataType="xs:string"<Value>False</Value>
</Attribute>
<Attribute Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Ports.P1R.EndOfSyncDomain"
AttributeDataType="xs:string"
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Ports.P1R.EndOfTopologyDiscovery"
AttributeDataType="xs:string>
<Value>False</Value>
</Attribute>
<Attribute Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Ports.P1R.PortActivation"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Ports.P1R.PortMonitoring"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute
Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Ports.P1R.TransmissionRateAndDuplex"
AttributeDataType="xs:string">
<Value>TransmissionRateAndDuplex.Automatic</Value>
</Attribute>
```

Node configuration

Following is the identifying name format for Node custom attribute in the AML file.

- <TypeIdentifier>.Interfaces.<InterfaceLabel>.Nodes.<NodeLabel>.<AttributeName> where

Custom attribute	Description
TypeIdentifier	Normalized TypeIdentifier (including FW version) of a pluggable (sub)module which is formatted to replace special characters with '_'.
AttributeName	The name of the attribute in Openness

Below snippet shows the format in which custom attributes shall be expected in the AML file if the Node is found over Interface.

```
...
<InternalElement ID="609cefa0-208f-4c51-bee5-60002332ef84" Name="E1">
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute
Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Nodes.Ethernet.PnDeviceNameAutoGeneration"
AttributeDataType="xs:string"><Value>True</Value>
</Attribute>
<Attribute
Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Nodes.Ethernet.PnDeviceNameSetDirectly"
AttributeDataType="xs:string"<Value>False</Value>
</Attribute>
<Attribute Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Nodes.Ethernet.UseIsoProtocol"
AttributeDataType="xs:string"><Value>False</Value>
</Attribute>
<Attribute Name="6ES7511_1AK00_0AB0_V1_8.Interfaces.X1.Nodes.Ethernet.UseRouter"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationEthernetRoleClassLib/
NodeEthernet" />
</InternalElement>
```

Subnet configuration

Following is the identifying name format for Subnet custom attribute in the AML file.

- <TypeIdentifier>.<AttributeName> where

Custom attribute	Description
TypeIdentifier	Normalized TypeIdentifier of a subnet which is formatted to replace special characters with '_'.
AttributeName	The name of the attribute in Openness

Below snippet shows the format in which custom attributes shall be expected in the AML file if the Subnet is found.

```
...
<InternalElement ID="cb471149-05a1-4063-b591-25a00d84c288" Name="PROFIBUS_1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Profibus</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="System_Subnet_Profibus.BusProfile" AttributeDataType="xs:string">
<Value>BusProfile.Dp</Value>
</Attribute>
<Attribute Name="System_Subnet_Profibus.HighestAddress" AttributeDataType="xs:string">
<Value>126</Value>
</Attribute>
<Attribute Name="System_Subnet_Profibus.PbCableConfiguration"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute Name="System_Subnet_Profibus.PbCyclicDistribution"
AttributeDataType="xs:string">
<Value>True</Value>
</Attribute>
<Attribute Name="System_Subnet_Profibus.SubnetId" AttributeDataType="xs:string">
<Value>A5B5-1</Value>
</Attribute>
<Attribute Name="System_Subnet_Profibus.TransmissionSpeed"
AttributeDataType="xs:string"><Value>BaudRate.Baud1500000</Value>
</Attribute>
</Attribute>
```

IO System configuration

Following is the identifying name format for IO Sysstem custom attribute in the AML file.

- <TypeIdentifier>.Interfaces.<InterfaceLabel>.IOSystem.<AttributeName> where

Custom attribute	Description
TypeIdentifier	Normalized TypeIdentifier (including FW version) of a pluggable (sub)module which is formatted to replace special characters with '_'.
AttributeName	The name of the attribute in Openness

Below snippet shows the format in which custom attributes shall be expected in the AML file if the IO System is found over Interface.

```
...
<InternalElement ID="f6edc86b-f995-4459-9429-b30c36b0b475" Name="PROFINET IO-System">
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>100</Value>
</Attribute>
<Attribute Name="CustomAttributes">
<RefSemantic CorrespondingAttributePath="ListType" />
<Attribute Name="6ES7515_2AM00_0AB0_V1_8.Interfaces.X1.IoSystem.MultipleUseIoSystem"
AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
<Attribute
Name="6ES7515_2AM00_0AB0_V1_8.Interfaces.X1.IoSystem.UseIoSystemNameAsDeviceNameExtension"
 AttributeDataType="xs:string">
<Value>False</Value>
</Attribute>
</Attribute>
```

Tolerant Import

During Import, TIA Portal is tolerant wrt custom attribute names at the following scenarios:

- CAx import accepts AML file with custom attributes having complete IdentifyingName.
- CAx import accepts AML file with custom attributes having only attribute name.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.10 Export/import AML file with pluggable port configuration

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

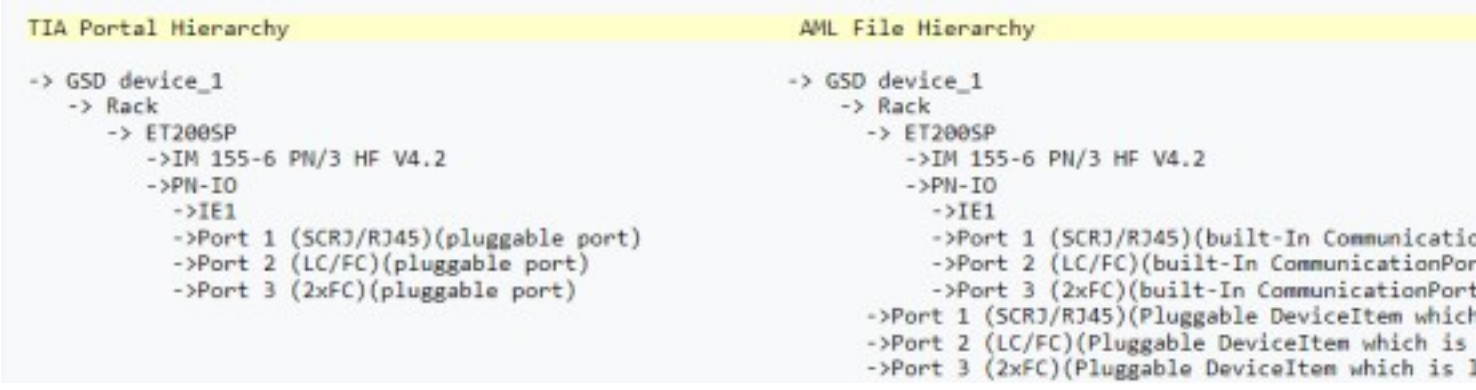
Introduction

In TIA Portal, devices can have built-in and pluggable ports configuration. While built-in ports can be represented in AML files without changes, whereas pluggable ports need a transformation since external tools like EPLAN does not support configuring/exchanging pluggable ports directly under a network interface.

Therefore, AR APC 1.4.0 has introduced a transformation approach that divides the original pluggable port into two parts (Pluggable DeviceItem and BuiltIn CommunicationPort) within the AML file. These parts are then connected using a new external interface type and an internal link.

The custom attributes will remain accessible under the pluggable DeviceItem after this transformation.

TIA Portal and AML hierarchy



Export

The AML file that will be generated during export for the above configuration is shown below.

```
<?xmlversion="1.0"encoding="utf-8"?>
<CAEXFilexmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"FileName="PluggablePortTransformations.aml"SchemaVersion="2.15"xsi:noNamespaceSch
emaLocation="CAEX_ClassModel_V2.15.xsd">.
/..
<InstanceHierarchyName="APCSampleInstanceHierarchy"><InternalElementID="7b00480c-5708-4ffa
-ac2d-d571b57be6af"Name="GSDdevice_1">
<AttributeName="TypeIdentifier"AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.33-SIEMENS-ET200SP-20180406.XML/D</Value>
</Attribute></InternalElement>
<InternalElementID="63dbb1a0-3b53-47b9-a38e-333b15a93f59"Name="Port1(SCRJ/RJ45)">
<AttributeName="Label"AttributeDataType="xs:string"><Value>P1R</Value>
</Attribute>
<AttributeName="PositionNumber"AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<AttributeName="BuiltIn"AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterfaceID="c3041983-aad3-41b6-a378-
c8c2f5dc2608"Name="CommunicationPortInterface"RefBaseClassPath="AutomationProjectConfigura
tionInterfaceClassLib/CommunicationPortInterface"/>
<ExternalInterfaceID="c738690b-a613-4bcb-bba7-
ed1c35c0f5a9"Name="CommunicationPortProxyInterface"RefBaseClassPath="AutomationProjectConf
igurationInterfaceClassLib/CommunicationPortProxyInterface"/
><SupportedRoleClassRefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort"/>
</InternalElement>
</InternalElement>
<InternalElementID="63ac85bc-b31f-4176-b9dd-894366458d72"Name="Port1(SCRJ/RJ45)">
<AttributeName="TypeName"AttributeDataType="xs:string">
<Value>Port1(SCRJ/RJ45)</Value>
</Attribute>
<AttributeName="PositionNumber"AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<AttributeName="BuiltIn"AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<AttributeName="TypeIdentifier"AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.33-SIEMENS-ET200SP-20180406.XML/SM/IDS_1P1HFFO_RJ45V4.3</Value>
</Attribute>
<AttributeName="InstallationDate"AttributeDataType="xs:dateTime">
<Value>2023-07-25T11:10:49.3048765Z</Value>
</Attribute>
<AttributeName="CustomAttributes">
<RefSemanticCorrespondingAttributePath="ListType"/
><AttributeName="GSD_GSDML_V2_33_SIEMENS_ET200SP_20180406_XML_SM_IDS_1P1_HF_FO_RJ45_V4_3.A
lternativePartnerPorts"AttributeDataType="xs:string">
<Value>False</Value></Attribute>
...
</Attribute>
<ExternalInterfaceID="0655ee61-7a4b-4ece-99a9cb626893f996"Name="CommunicationPortProxyInte
rface"RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortProxyInterface"/
```

```
><SupportedRoleClassRefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem"/>
</InternalElement>
</InternalElement>
<SupportedRoleClassRefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem"/>
<InternalLinkName="LinkToProxyPort_1"RefPartnerSideA="63ac85bc-b31f-4176-
b9dd-894366458d72:CommunicationPortProxyInterface"RefPartnerSideB="63dbb1a0-3b53-47b9-
a38e-333b15a93f59:CommunicationPortProxyInterface"/>
<InternalLinkName="LinkToProxyPort_2"RefPartnerSideA="156250d0-f6f6-4bc7-97c6-
e910455c78ae:CommunicationPortProxyInterface"RefPartnerSideB="79183be2-9768-4f32-
a2ff-9ece9f65a64e:CommunicationPortProxyInterface"/>
</InternalElement>
<SupportedRoleClassRefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem"/>
</InternalElement>
<SupportedRoleClassRefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device"/>
</InternalElement>
<SupportedRoleClassRefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceUserFolder"/>
</InternalElement>
<SupportedRoleClassRefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
AutomationProject"/>
</InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

Port and DeviceItem connection-

The XML representation of Port-DeviceItem connections between multiple ports and DeviceItems will be done using format shown below.

- ExternalInterface-<ExternalInterface> of type "CommunicationPortProxyInterface" will be added under <CommunicationPort> and <DeviceItem> internal element which participates in the connection.

```
<InternalElement ID="[Port Unique ID]" Name="[Port Name]">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="[External Interface Unique ID]"
Name="CommunicationPortProxyInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortProxyInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="[DeviceItem Unique ID]" Name="[DeviceItem Name]">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Port 2 (LC/FC)</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.33-SIEMENS-ET200SP-20180406.XML/SM/IDS_1P2 HF LC_FC V4.3</Value>
</Attribute>
<ExternalInterface ID="c7066655-8895-4969-ae44-4533ad5319d7"
Name="CommunicationPortProxyInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortProxyInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

- Internal link- Port-DeviceItem connections are represented using <InternalLink> tags. <InternalLink> tags shall be added under common parent of multiple port and deviceItems(i.e., Rack). Internal link name shall be unique across the common parent.

```
<InternalLink Name="Link To [Internal link Name]" RefPartnerSideA="[Communication Port
UniqueID]:CommunicationPortProxyInterface" RefPartnerSideB="[DeviceItem
UniqueID]:CommunicationPortProxyInterface" />
```

Import

It shall be possible to import transformed pluggable port details from an AML file which is generated out of above mentioned export. However, it shall also be possible to import AML files which are created in previous versions of TIA Portal project.

Note

- The export hierarchy change behavior shall be applicable only in version V19 onwards. Older version TIA portal shall have same behavior as previous.
- The hierarchy in AML file shall not influence/affect the TIA Portal internal hierarchy after import.
- This hierarchy change/transformation behavior is applicable for any pluggable port configuration(Scalance devices, ET200SP..).

6.5.11 Export/Import AML file with IO Link

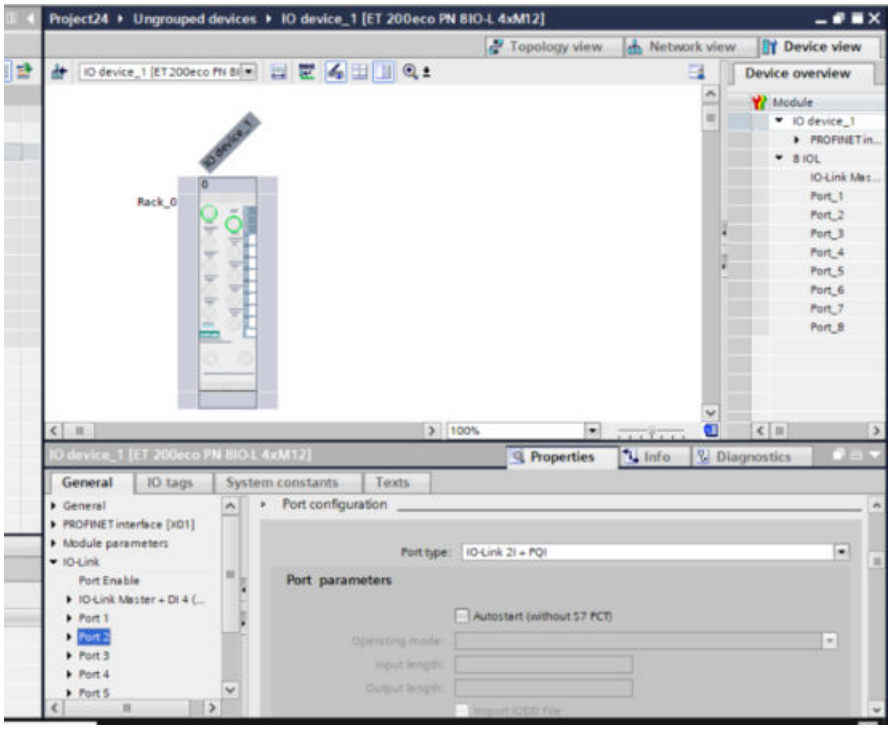
Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

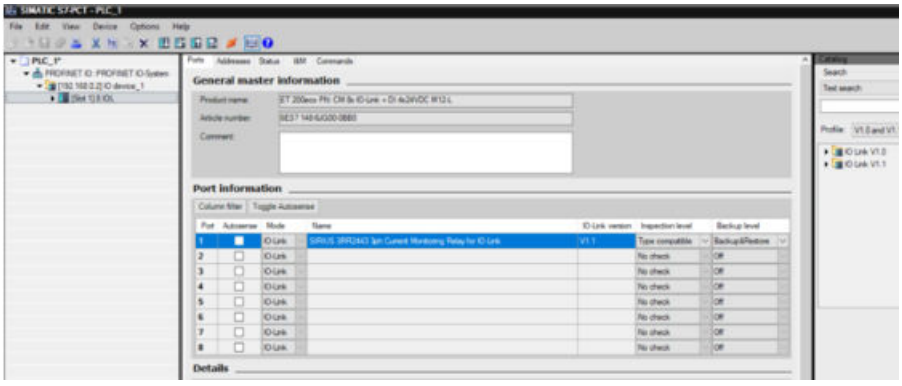
Introduction

You can use the TIA Portal to import and export an AML file containing IO Link master configuration along with IO Link devices configured via S7-PCT (Port Configuration Tool).

Below is a simple configuration example with one IO Link master:



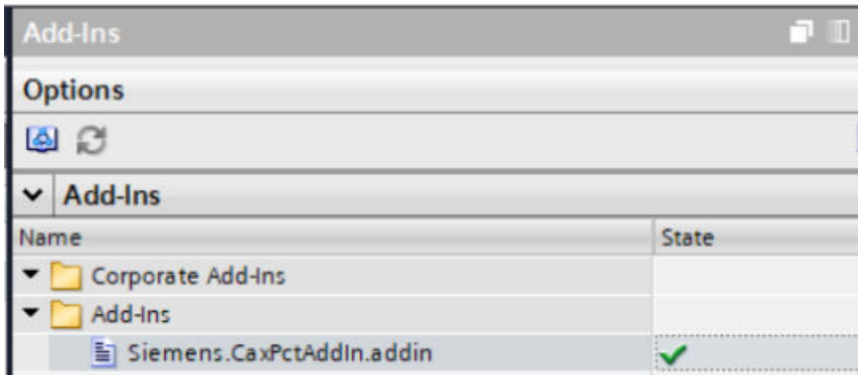
Below is configuration example of IO Link Device Configured on IO Master Port via PCT:



Activating/Deactivating Add-In

For export/import of IO Link master configuration(configured using PCT in TIA Portal) via AML file, Siemens.CaxPctAddIn.addin needs to be activated from TIA Portal user interface.

You should be able to activate/deactivate Add-In from user interface. For Add-In activation request, you shall be prompted with required general and workflow permission access.



Note

'Siemens.CaxPctAddIn.addin' file needs to be copied from "Setup\\Bundles\ \WinCCCA_PRO_DVD_2\\DVD\\Support\\CAx\\Add-Ins" to enable activation/de-activation of Add-in.

Export AML file with IO Link information

You can use the TIA Portal to export project configuration containing IO Link masters along with connected IO Link devices that are configured via PCT. On successful export, you shall be able to see the same configuration in AML file. If Add-In is not activated, export should still be successful, but IO Link devices related configuration shall not be exported.

If there are any issues with exporting PCT relevant configuration, issue details should be logged as appropriate error/warning in PCT.log file. Either PCT export is successful or not, a general appropriate message shall be shown in TIAP UI Info tab.

PCT log file location : PCT log file created in same location as of CAX log file <CAX log location>\<guid>\PCT.log".

- If IO Link master is configured without S7 PCT (For example: ports are having Configuration without S7-PCT = true in TIA Portal) because you have decided to configure IO Link Ports via TIA Portal, ports are still exported from TIA Portal but with minimum attributes like ConfigurationWithPDCT, Label and Name in AML file and same shall be imported back during AML file.
- There are some configuration (like HW parameters/attributes) in TIA Portal which can influence IO Link port configuration on PCT side. For example: In module 6ES7 148-6JG00-0BB0, every port can be configured with "port type" in TIA Portal which influences on type of the IO Link devices can be plugged to that port on PCT side. This parameter controls required address space allocation for every port on PCT side to plug required IO Link device. During AML exchange, you should ensure to include such 'influencing' configuration in AML file so that AML exchange round trips are successful. When such 'influencing' configuration is missing in the AML file, AML round trip shall not be successful. For the example mentioned above, user can include custom attributes export (via CAx settings in TIA Portal), so that 'port type' parameter shall be part of AML during export from TIA Portal.

- An user is always expected to manually sync the IO Link configuration changes done in TIA Portal by explicitly launching the S7-PCT tool. So, it shall be user's responsibility to keep the IO-Link configuration up-to-date before exporting the AML file. For example: Initially, user creates an IO Link configuration by selecting default parameters for ports in TIA Portal side and then configures IO Link devices on S7-PCT. Later, user decides to change parameters for ports in TIA Portal side. Once user changes, user shall make sure to launch S7-PCT tool so that any inconsistencies that would have occurred due to parameter change (like earlier IO Link device plugged does not fit to port) would be notified and same shall be resolved by the user before AML export. Otherwise, inconsistent configuration may be exported.
- There are configurations where IO Link master ports can be configured initially without S7-PCT (For ex: setting Operating mode = IO-Link Manual) but can later be changed/overridden in S7-PCT tool. This configuration shall not be supported by AML exchange.

Below is the snippet of AML created after exporting the configuration via CAx export:

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd"
FileName="Project2_.aml" SchemaVersion="2.15">
..
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="02afa376-7fbd-46d8-9659-b82959617326" Name="Project24">
..
<InternalElement ID="41fec27b-a0fa-4df6-a604-d60008e34d81" Name="Ungrouped devices">
<InternalElement ID="968336bc-bda8-4da6-b3fa-2cc8e8493806" Name="ET 200eco station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.ET200eco</Value>
</Attribute>
<InternalElement ID="41196a85-aaae-42e0-994f-4a46e773af0f" Name="Rack_0">
...
<InternalElement ID="165b1850-bba5-4509-af92-acafd8f4f1d1" Name="IO device_1">
...
<InternalElement ID="555d1dd7-ec4d-40e6-b34d-06443a255741" Name="IOLink">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>IO-Link</Value>
</Attribute>
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>IO-Link</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="4670a2aa-f441-458d-8dcc-e31b7712e325" Name="IOLink Port 1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>Port 1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute><Attribute Name="ConfigurationWithPDCT" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
..
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" /><SupportedRoleClass
RefRoleClassPath="AutomationProjectConfigurationIOLinkRoleClassLib/
CommunicationPortIOLink" />
</InternalElement>
..
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" </InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem">
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationIOLinkRoleClassLib/
DeviceItemIOLinkMaster" />
</InternalElement>
..
```

```
</InternalElement>
<InternalElement ID="73292b01-7f22-44c8-98b1-2a007580cea0" Name="SIRIUS 3RR2441 3ph
Current Monitoring Relay for IO-Link">
..
<InternalElement ID="d55d0b10-0d2f-44e6-9b96-72725a88ca41" Name="SIRIUS 3RR2441 3ph
Current Monitoring Relay for IO-Link">
..
<InternalElement ID="ab97d538-5e44-4a97-bab1-d228c1af8c0b" Name="IOLink">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>IO-Link</Value>
</Attribute>
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>IO-Link</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="f6ff4404-58fb-43ce-bf5d-e365e55c17b5" Name="IOLink Port 1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>Port 1</Value>
</Attribute>
<ExternalInterface ID="065e309d-01cc-410e-951d-0fa7cac979d0"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" /
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationIOLinkRoleClassLib/
CommunicationPortIOLink" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" /
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationIOLinkRoleClassLib/
DeviceItemIOLinkDevice" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
AutomationProject" />
..
<InternalLink Name="IOLink Port 1" RefPartnerSideA="f6ff4404-58fb-43ce-bf5d-
e365e55c17b5:CommunicationPortInterface" RefPartnerSideB="4670a2aa-f441-458d-8dcc-
e31b7712e325:CommunicationPortInterface" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

Import AML file with IO Link information

You should be able to import AML file containing IO Link master and IO Link device configuration that can be configured via PCT. On a Successful import, you would be able to see the IO Link devices connected to IO Link master ports after opening PCT tool. If Add-In is not activated, import should still be successful but IO Link devices related configuration will not be imported.

If there are any issues with importing PCT relevant configuration, issue details should be logged as appropriate error/warning in PCT.log file. Either PCT import is successful or not, a general appropriate message shall be shown in TIA Portal user interface Info tab.

Whitelist

Following modules shall support export/import IO Link master configuration.

	IO Link Master	Firmware Version
ET 200SP	6ES7-137-6BD00-0BA0	2.2
ET 200AL	6ES7-147-5JD00-0BA0	1.2
	6ES7-147-5JD00-0BA0	1.1
ET 200pro	6ES7-147-4JD00-0AB0	1.1
ET 200MP	6ES7-547-1JF00-0AB0	1.0
ET 200eco PN	6ES7-148-6JD00-0AB0	1.1
	6ES7-148-6JD00-0AB0	1.0
	6ES7-148-6JG00-0BB0	5.1
	6ES7-148-6JG00-0BB0	1.1
	6ES7-148-6JG00-0BB0	1.0
	6ES7 148-6JE00-0BB0	5.1
	6ES7 148-6JJ00-0BB0	5.1
SIPLUS ET 200SP	6AG1-137-6BD00-2BA0	2.2
	6AG2-137-6BD00-1BA0	2.2

Blacklist

Following modules shall not support export/import IO Link master configuration.

	IO Link Master	Firmware Version
ET 200S	6ES7-138-4GA50-0AB0	
	3RK1-005-0LB00-0AA0	1.0
ET 200SP	6ES7-137-6BD00-0BA0	2.1
	6ES7-137-6BD00-0BA0	2.0
	6ES7-137-6BD00-0BA0	1.0
ET 200AL	6ES7-147-5JD00-0BA0	1.0
ET 200pro	6ES7-147-4JD00-0AB0	1.0
ET 200eco PN	6ES7-148-6JA00-0AB0	7.0
	6ES7-148-6JA00-0AB0	6.1

	IO Link Master	Firmware Version
S7-1200	6ES7-278-4BD32-0XB0	2.1
	6ES7-278-4BD32-0XB0	2.0
SIPLUS S7-1200	6AG1-278-4BD32-2XB0	2.1
	6AG1-278-4BD32-2XB0	2.0
	6AG1-278-4BD32-4XB0	2.1
	6AG1-278-4BD32-4XB0	2.0
SIPLUS ET 200SP	6AG1-137-6BD00-2BA0	2.1
	6AG2-137-6BD00-1BA0	2.1

See also

Opening a project (Page 128)

6.5.12 Connection handling for extension racks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

You can use the TIA Portal Openness to fetch, add, and remove extension rack connections so that you can make use of TIA Portal Openness to implement extension rack connection support during CAx export/import.

Program code

```
ImConnection imConnection = portDeviceItem.GetService<ImConnection>();
imConnection.Connect(partnerport);
imConnection.Disconnect();
imConnection.GetPartnerPort();
var imConnectionOwner = imConnection.OwnedBy;
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.13 Export/Import AML file with complex tag configuration

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- You have opened a project via a TIA Portal Openness application
  See Opening a project (Page 128)

Application

You can use the TIA Portal to import an AML file containing Complex tags with User Defined Datatypes. Complex tags are created on TIA Portal which are present in mentioned tag table in AML file, however user defined data type would not be created. You need to manually create the required UDT.

Logical address of a complex is calculated from sub tags of a Complex tag. From list of all sub tags of a complex tag, the tag which have lowest minimum address will be picked and address of that tag will be assigned to Parent complex tag. If any sub tag has in valid address, the address of parent complex tag will not be calculated and set to empty.

On a successful import, you would be able to see all the valid complex tags created on TIA Portal with valid logical address. Export wise there are no changes and only the tags are exported.

AML file with user defined data type

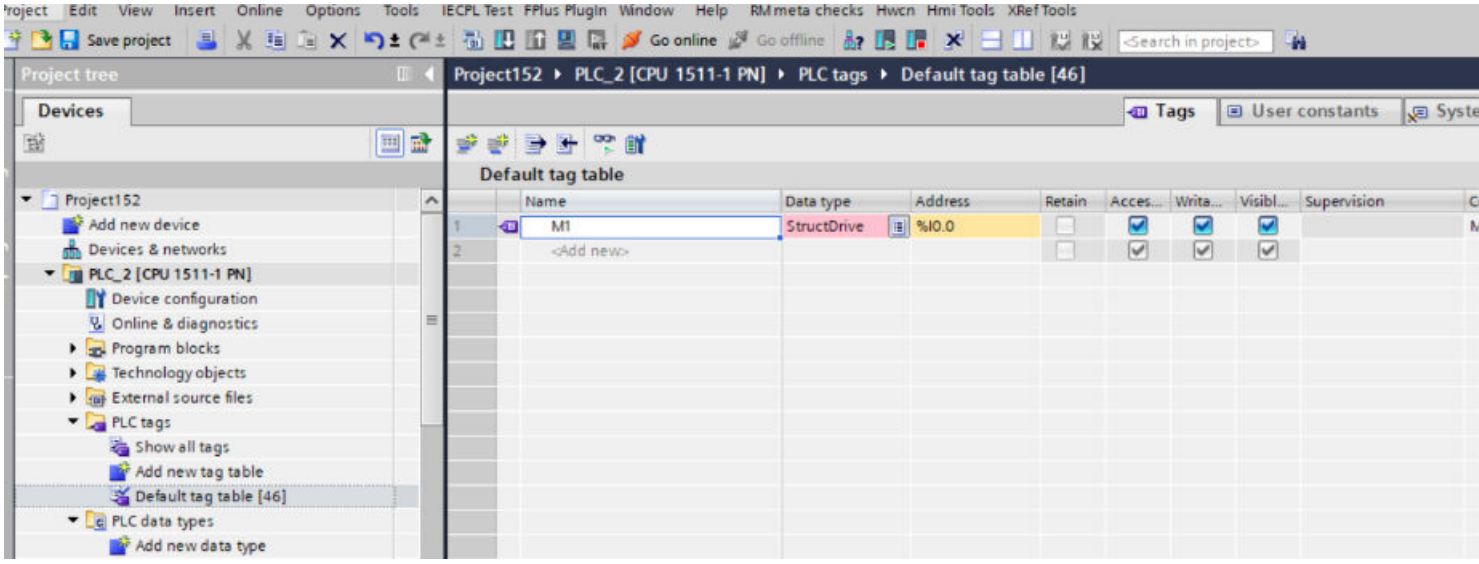
Below is a sample AML file containing a complex tag with user defined data type.

```
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" FileName="singletag.aml"
SchemaVersion="2.15" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
...
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="debc8e5c-2cbb-4fa9-afaa-de242e706dff" Name="Project97">
<InternalElement ID="dc142279-d745-432b-8451-4b282b55b3e5" Name="S71500/ET200MP
station_2">
...
<InternalElement ID="eedd64da-22f0-4b6e-babe-75e09e5cfc46" Name="Rail_0">
...
<InternalElement ID="a6e21bd8-f58c-4920-89ee-588d0eb34645" Name="PLC_2">
...
<InternalElement Name="TagTable" ID="065F26B8-E5BA-4BF3-A210-4E26E19F8A30">
<Attribute Name="AssignToDefault" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement Name="M1" ID="4AF292AA-DF8E-4A7A-AC9F-005B4DED47A5">
<Attribute Name="DataType" AttributeDataType="xs:string">
<Value>StructDrive</Value>
</Attribute>
<ExternalInterface Name="On"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag"
ID="5FCB82C1-019C-4833-A1DF-16721A5239FC">
<Attribute Name="DataType" AttributeDataType="xs:string">
<Value>BOOL</Value>
</Attribute>
<Attribute Name="LogicalAddress" AttributeDataType="xs:string">
<Value>I0.0</Value>
</Attribute>
<Attribute Name="Comment" AttributeDataType="xs:string">
<Value>Motor drive on</Value>
<Attribute Name="aml-lang=de-DE" AttributeDataType="xs:string">
<Value>Motor drive on</Value>
</Attribute>
</Attribute>
</ExternalInterface>
<ExternalInterface Name="Overheat"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag"
ID="331659A8-147B-407A-9629-5D85AE04890D">
<Attribute Name="DataType" AttributeDataType="xs:string">
<Value>BOOL</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
<Attribute Name="LogicalAddress" AttributeDataType="xs:string">
<Value>E0.1</Value>
</Attribute>
<Attribute Name="Comment" AttributeDataType="xs:string">
<Value>Motor drive overheat</Value>
<Attribute Name="aml-lang=de-DE" AttributeDataType="xs:string">
<Value>Motor drive overheat</Value>
```

```
</Attribute>
</Attribute>
</ExternalInterface>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
ComplexTag" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
TagTable" />
</InternalElement>
<InternalElement ID="5983e7fd-ca6a-4573-8579-9ff323cfa09a" Name="PROFINET interface_1">
...
<InternalElement ID="2240a28c-ba88-4af4-ae6a-03d0ba3ce108" Name="E1">
...>
</InternalElement>
<InternalElement ID="df1d95d3-89ee-4a45-bca8-73e342497bee" Name="Port_1">
...
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="25f8dc3c-621d-4659-bb04-2d923df832c4" Name="Port_2">
...
</InternalElemen<SupportedRoleClass
RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" /></InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
AutomationProject" /></InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

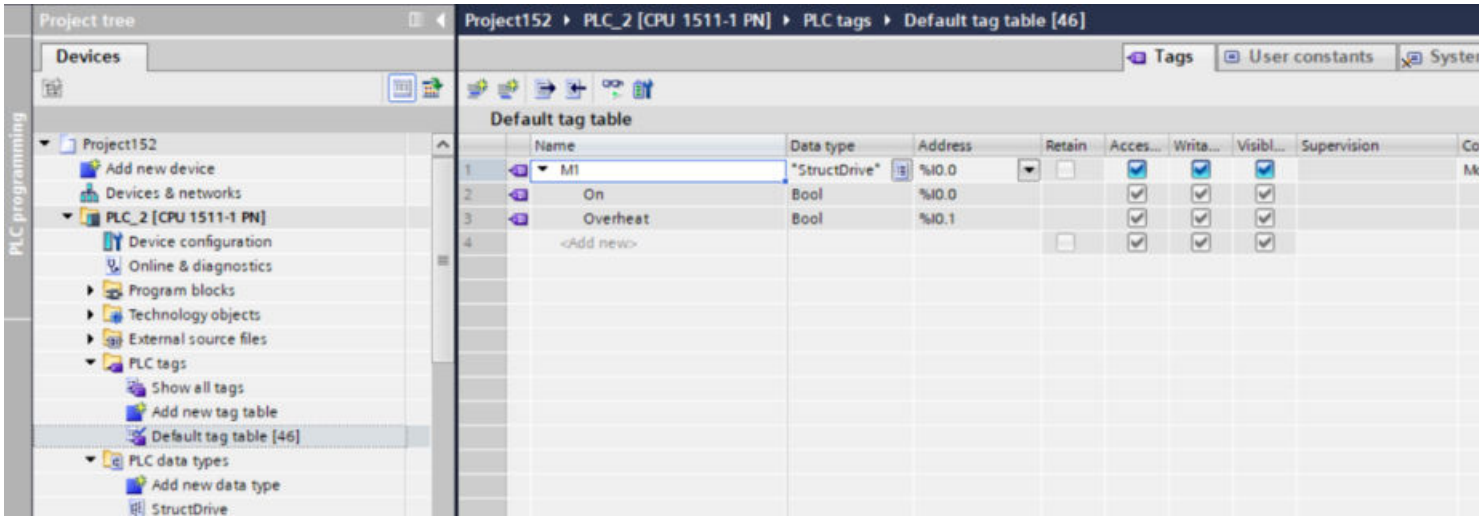
Complex Tag Import on TIA Portal Without UDT Creation

The above file has a complex tag which has two sub tags 'On' and 'Overheat'. After import into TIA Portal, complex tag shall be created like below:

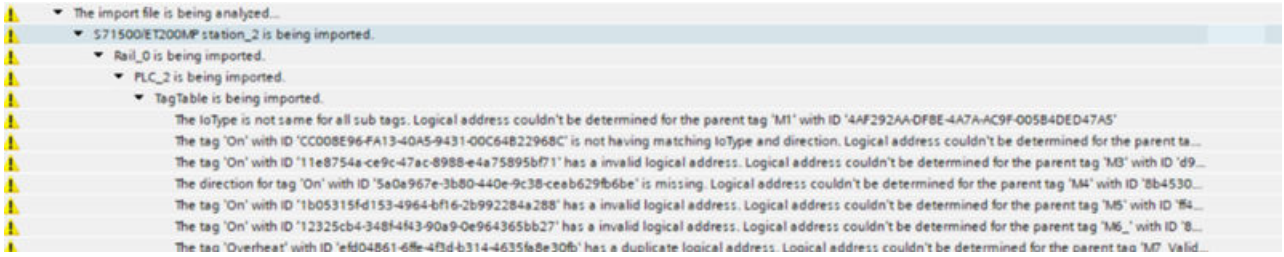


Complex Tag Import on TIA Portal with UDT Creation

An error indication on 'Data type' shall be shown incase datatype is missing or invalid. In the above case it is due to missing user defined datatype i.e, StructDrive.
Once you create the user defined datatype(StructDrive), the complex tags would look like below:



If there are any issue with any of the sub tags like their logical address is missing or any issues related to Direction, Iotype etc, it shall not be possible to determine lowest address of complex tag. In this case, complex tags shall be created without logical address and appropriate warning message shall be shown to the user like below:



See also

Connecting to the TIA Portal (Page 82)

6.5.14 Export of CAx data

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Introduction

In TIA Portal you can export your configuration in the device&networks editor to an AML file. This function is based on TIA Portal Openness and enables you to export hardware data from project or device level.

You can use the Export function in TIA Portal Openness to export CAx data: The export function can be accessed via CaxProvider. You can invoke the GetService method at Project object to get the CaxProvider service.

With TIA Portal V17 user interface, It shall be possible to perform CAx export and import operation in multiuser environment. If a Server project is opened alongside the local session, then Export and Import will happen for Server Project. For exporting and importing local session, the Server project needs to be closed.

Furthermore, for purpose of single device export as CAx, device context menu shall have menu entry for performing Export for a Server project in case it is open. In this case the context menu shall not appear for local Session. For a single device export as CAx, the context menu shall be available for a local session only when user closes Server project view.

It shall also be possible in TIA Portal Openness V17 to perform CAx export and import operation in multi user environment. For multiuser project (local session and server project) to participate in exchange via AML, a new API for export is used. The new Export API takes 'ProjectBase' as an argument. You can now use Export API to export both single and multiuser project.

As of TIA Portal OpennessV17, CAx export/import functionality through UI and API will no longer check:

- Windows Group for Openness 'Siemens TIA Openness' is present
- Windows User is a member of 'Siemens TIA Openness' Group

Note

For invoking CAx APIs in a client application, you have to make sure that either the application is signed with proper Openness certificate, or Openness 'Siemens TIA Openness' Group is present and the user is a member of the group.

With TIA Portal V19, You can access the result messages of CAx Export and Import operation programmatically, so that you are able to have custom analysis and visualization of the transfer results. The result contains the messages with the information.

The new Export API for device and project provide a structured TransferResult as return value and the log file is not being generated. You can process the transfer result and store it in a custom format.

Export and Import restrictions for CAx

CAx will not support export and import of following devices:

- HMI devices except push button panels and key panels
- Drives

CAx will not support export and import of certain Scalance device items in TIA Portal.

- 6GK5 602-0BA10-2AA3
- 6GK5 612-0BA10-2AA3
- 6GK5 623-0BA10-2AA3
- 6GK5 627-2BA10-2AA3
- 6GK5 602-0BA00-2AA3
- 6GK5 612-0BA00-2AA3
- 6GK5 613-0BA00-2AA3
- System:Device.Scalance/S627
- 6GK5 495-8BA00-8AA2
- 6ES7 451-3AL00-0AE
- 6AV2 104-0xxxx-xxxx
- 6AV2 155-xxxxx-xxxx
- System:Rack.Scalance/X200BA-Y V4.2
- System:PC.DCSPlus.LogicalServer
- System:Device.DCS_ReduMCStation
- System:Device.DCS_MCStation

- System:Device.DCS_SimulationStation
- System:Device.OPCUAPackageUnit
- System:Device.IEC60870
- System:Device.S7Com
- System:Device.CS3000

Note

From TIA Portal Openness V17 onwards, Normalized format of TypeIdentifier of above mentioned device items can also be considered for exclusion.

Program code: Access the `CaxProvider` service

Modify the following program code to access the `CaxProvider` service:

```
//For Single Project
ProjectBase project = tiaPortal.Projects.Open(...);
//Or
//for LocalSession of a Multiuser Project
MultiuserProject project = tiaPortal.LocalSessions.Open(...).Project;
//Or
//for Server Project of a Multiuser Project
MultiuserProject project = tiaPortal.LocalSessions.OpenServerProject(...).Project;
Or
//Single user Project
Project project = tiaPortal.Projects.Open(...);
CaxProvider caxProvider = project.GetService<CaxProvider>();
if(caxProvider != null)
{
// Perform CAx export and import operation
}
```

CAx export at project level

To export CAx data at project level, you can use the `Export()` with the following parameters in TIA Portal V14 onwards:

Name	Type	Example	Description
ProjectToExport	Project	tiaPortal.Projects[0]	Project object to Export
ExportFilePath	FileInfo	new FileInfo(@"D:\Temp\ProjectExport.aml")	Full Export file path of AML file
LogFilePath	FileInfo	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	Full file path of Log file

To export CAx data at project level, use the Export() with the following parameters in TIA Portal V17 onwards:

Name	Type	Description
ProjectToExport	ProjectBase	Single or Multiuser Project object to Export
ExportFilePath	FileInfo	Full Export file path of AML file
LogFilePath	FileInfo	Full file path of Log file

Return Type

Type	Description
bool	True, if CAx Export operation completed without errors; otherwise False

Modify the following program code to export CAx data at project level:

```
caxProvider.Export(project, new FileInfo(@"D:\Temp\ProjectExport.aml"), new
FileInfo(@"D:\Temp\ProjectExport_Log.log"));
```

CAx export overload API at project level

To export CAx data at project level with an access to the result messages, you can use the 'Export' overload API with the following parameters in TIA Portal V19 onwards::

Name	Type	Description
projectToExport	ProjectBase	Single or Multiuser Project object to Export
exportFilePath	FileInfo	Full Export file path of AML file

Return Type

Type	Description
TransferResult	Result of a CAx transfer.

The following properties are supported in TransferResult :

Property name	Type	Access
ErrorCount	int	Read
WarningCount	int	Read
State	TransferResultState	Read
Messages	TransferResultMessageComposition	Read

The following properties are supported in TransferResultMessageComposition:

Property name	Type	Access
this[int]	TransferResultMessage	Read

The following properties are supported in TransferResultMessage:

Property name	Data type	Access
DateTime	DateTime	Read
ErrorCount	Int	Read
WarningCount	Int	Read
State	TransferResultState	Read
Message	String	Read

The list of possible transfer result states:

Enum option	Description
TransferResultState.Success	Transfer finished successfully
TransferResultState.Information	Transfer finished with information
TransferResultState.Warning	Transfer finished with warnings
TransferResultState.Error	Transfer finished with errors

Modify the following program code to export CAx data at project level:

```
private static void CaxTransferAtProjectLevel(Siemens.Engineering.ProjectBase project,
CaxProvider caxProvider)
{
FileInfo exportFilePath = new FileInfo("D:\\temp\\ExportFile.aml");
// New Export API for project:
TransferResult projectExportResult = caxProvider.Export(project, exportFilePath);
PrintCaxResult(projectExportResult);
}
private static void PrintCaxResult(Siemens.Engineering.Cax.TransferResult result)
{
Console.WriteLine($"CAx result summary: {result.State} (errors: {result.ErrorCount},
warnings: {result.WarningCount})");
PrintCaxDetailResult(result.Messages);
}
private static void
PrintCaxDetailResult(Siemens.Engineering.Cax.TransferResultMessageComposition messages,
int nestingDepth = 0)
{
foreach (Siemens.Engineering.Cax.TransferResultMessage message in messages)
{
string indent = new string(' ', nestingDepth * 2);
Console.WriteLine($"{indent}{message.State} {message.Message} {message.DateTime} (errors:
{message.ErrorCount}, warnings: {message.WarningCount})");
PrintCaxDetailResult(message.Messages, nestingDepth + 1);
}
}
```

CAx export at device level

To export CAx data at device level, use the `Export` method with the following parameters:

Name	Type	Example	Description
DeviceToExport	Device	project.Devices[0]	Device object to Export
ExportFilePath	FileInfo	new FileInfo(@"D:\Temp\ProjectExport.aml")	Full Export file path of AML file
LogFilePath	FileInfo	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	Full file path of Log file

Return Type

Type	Description
bool	True, if CAx Export operation completed without errors; otherwise False

Modify the following program code to export CAx data at device level:

```
caxProvider.Export(device, new FileInfo(@"D:\Temp\DeviceExport.aml"), new
FileInfo(@"D:\Temp\DeviceExport_Log.log"));
```

CAx export overload API at device level

To export CAx data at device level with an access to the result messages, you can use the Export overload API with the following parameters in TIA Portal V19 onwards:

Name	Type	Example	Description
deviceToExport	Device	project.Devices[0]	Device object to Export
exportFilePath	FileInfo	new FileInfo(@"D:\Temp\ProjectExport.aml")	Full Export file path of AML file

Return Type

Type	Description
TransferResult	Result of a CAx transfer.

The following properties are supported in TransferResult :

Property name	Type	Access
ErrorCount	int	Read
WarningCount	int	Read
State	TransferResultState	Read
Messages	TransferResultMessageComposition	Read

The following properties are supported in TransferResultMessageComposition:

Property name	Type	Access
this[int]	TransferResultMessage	Read

The following properties are supported in TransferResultMessage:

Property name	Data type	Access
DateTime	DateTime	Read
ErrorCount	Int	Read
WarningCount	Int	Read
State	TransferResultState	Read
Message	String	Read

The list of possible transfer result states:

Enum option	Description
TransferResultState.Success	Transfer finished successfully
TransferResultState.Information	Transfer finished with information
TransferResultState.Warning	Transfer finished with warnings
TransferResultState.Error	Transfer finished with errors

Modify the following program code to export CAx data at device level:

```
private static void CaxTransferAtDeviceLevel(Siemens.Engineering.ProjectBase project,
CaxProvider caxProvider)
{
FileInfo exportFilePath = new FileInfo("D:\\temp\\ExportFile.aml");
Device deviceToExport = project.Devices.Find("Station_1");
// New Export API for project:
TransferResult deviceExportResult = caxProvider.Export(deviceToExport, exportFilePath);
PrintCaxResult(deviceExportResult);
}
private static void PrintCaxResult(Siemens.Engineering.Cax.TransferResult result)
{
Console.WriteLine($"CAx result summary: {result.State} (errors: {result.ErrorCount},
warnings: {result.WarningCount})");
PrintCaxDetailResult(result.Messages);
}
private static void
PrintCaxDetailResult(Siemens.Engineering.Cax.TransferResultMessageComposition messages,
int nestingDepth = 0)
{
foreach (Siemens.Engineering.Cax.TransferResultMessage message in messages)
{
string indent = new string(' ', nestingDepth * 2);
Console.WriteLine($"{indent}{message.State} {message.Message} {message.DateTime} (errors:
{message.ErrorCount}, warnings: {message.WarningCount})");
PrintCaxDetailResult(message.Messages, nestingDepth + 1);
}
}
```

6.5.15 Import of CAx data

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)

Application

In TIA Portal you can import your configuration in the device&networks editor from an AML file. This function enables you to import hardware data from project or device level.

You can use the import function to import CAx data. The import function is accessed via `CaxProvider` service. You can invoke the GetService method at Project object to get the CaxProvider service.

With TIA Portal V19, the new import API provides a structured TransferResult as return value and the log file is not being generated. You can process the transfer result and store it in a custom format.

Program code: Access the `CaxProvider` service

Modify the following program code:

```
//Access the CaxProvider service
Project project = tiaPortal.Projects.Open(...);
CaxProvider caxProvider = project.GetService<CaxProvider>();
if(caxProvider != null)
{
      // Perform Cax export and import operation
}
```

CAx import

To import CAx data into a TIA Portal project, you can use the `Import` method with the following parameters:

Name	Example	Description
ImportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	Full import file path of AML file
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	Full file path of log file
ImportOptions	CaxImportOptions.MoveToParkingLot CaxImportOptions.RetainTiaDevice CaxImportOptions.OverwriteTiaDevice	Conflict resolution strategies in case of importing into an already existing non empty project.

Modify the following program code to import CAx data:

```
caxProvider.Import(new FileInfo(@"D:\Temp\ProjectImport.aml"), new
FileInfo(@"D:\Temp\ProjectImport_Log.log"), CaxImportOptions.MoveToParkingLot);
```

The following `CaxImportOptions` are provided:

Import option	Description
MoveToParkingLot	Retain name conflicting device/s in the project and import those out of CAx into a parkinglot folder
RetainTiaDevice	Retain name conflicting device/s in the project and do not import those out of CAx
OverwriteTiaDevice	Overwrite name conflicting device/s in the project by the ones out of CAx

From TIA Portal V18, CAx Import shall display an error message when product license(s) are missing and you try to import configuration which requires license for creation. And, the configuration which do not enforce product license check, shall be created without any error.

For example: The import of modules like S7-1500 PLCs (configured with TypeIdentifier means Article number in AML file) shall fail if 'Step7' product license missing whereas import of

Digital Input/Output modules (configured with TypeIdentifier means Article number in AML file) shall succeed.

Also, import of configuration based on 'TemplateIdentifier' in AML file shall be imported successfully as "Library" based configuration does not enforce product license check.
For example: The import of any modules like S7-1500 PLCs, Digital Input/Output modules etc., (when configured with TemplateIdentifier in AML file) shall succeed even when 'Step7' product license missing.
Missing license has no impact on CAx export. So, export shall behave like before.

CAx import overload API

To import CAx data into a TIA Portal project with an access to the result messages, you can use the Import overload API with the following parameters in TIA Portal V19 onwards:

Name	Example	Description
ImportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	Full import file path of AML file
ImportOptions	CaxImportOptions.MoveToParkingLot CaxImportOptions.RetainTiaDevice CaxImportOptions.OverwriteTiaDevice	Conflict resolution strategies in case of importing into an already existing non empty project.

Return Type

Type	Description
TransferResult	Result of a a CAx transfer

The following properties are supported in TransferResult type :

Property name	Type	Access
ErrorCount	int	Read
WarningCount	int	Read
State	TransferResultState	Read
Messages	TransferResultMessageComposition	Read

The following properties are supported in TransferResultMessageComposition:

Property name	Type	Access
this[int]	TransferResultMessage	Read

The following properties are supported in TransferResultMessage:

Property name	Data type	Access
DateTime	DateTime	Read
ErrorCount	Int	Read
WarningCount	Int	Read
State	TransferResultState	Read
Message	String	Read

The list of possible transfer result states:

Enum option	Description
TransferResultState.Success	Transfer finished successfully
TransferResultState.Information	Transfer finished with information
TransferResultState.Warning	Transfer finished with warnings
TransferResultState.Error	Transfer finished with errors

Modify the following program code to import CAx data:

```
private static void ImportCaxTransfer(Siemens.Engineering.ProjectBase project, CaxProvider
caxProvider)
{
FileInfo importFilePath = new FileInfo("D:\\temp\\ImportFile.aml");
CaxImportOptions importOption = Siemens.Engineering.Cax.CaxImportOptions.RetainTiaDevice;
// New Import API:
TransferResult importResult = caxProvider.Import(importFilePath, importOption);
PrintCaxResult(importResult);
}
private static void PrintCaxResult(Siemens.Engineering.Cax.TransferResult result)
{
Console.WriteLine($"CAx result summary: {result.State} (errors: {result.ErrorCount},
warnings: {result.WarningCount})");
PrintCaxDetailResult(result.Messages);
}
private static void
PrintCaxDetailResult(Siemens.Engineering.Cax.TransferResultMessageComposition messages,
int nestingDepth = 0)
{
foreach (Siemens.Engineering.Cax.TransferResultMessage message in messages)
{
string indent = new string(' ', nestingDepth * 2);
Console.WriteLine($"{indent}{message.State} {message.Message} {message.DateTime} (errors:
{message.ErrorCount}, warnings: {message.WarningCount})");
PrintCaxDetailResult(message.Messages, nestingDepth + 1);
}
}
```

6.5.16 Export/Import of sub modules

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- PLC is offline

Application

You can have round trip exchange of sub modules data between the TIA Portal and other engineering tools, e.g. CAD tool like EPLAN by keeping a common hierarchy for sub modules inside AML file during export and import. For example, the sub modules like Bus Adapters shall have different internal hierarchy in TIA Portal than in other applications (e.g., CAD tools like EPLAN).

AML structure of the export file

You can export sub modules data from TIA Portal hierarchy to AML file hierarchy.

The following example depicts the partial AML file structure that shall be generated during the export of Bus Adapter as sub module from TIA Portal .

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile FileName="Project4.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
<WriterName>Totally Integrated Automation Portal</WriterName>
<WriterID>1d4fcebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
<WriterVendor>Siemens AG</WriterVendor>
<WriterVendorURL>www.siemens.com</WriterVendorURL>
<WriterVersion>15</WriterVersion>
<WriterRelease>1500.0100.0.0</WriterRelease>
<LastWritingDateTime>2018-05-03T11:23:10.3011329Z</LastWritingDateTime>
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="6cd7f80f-e049-4958-ba67-630481805bf0" Name="Project4">
<Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
<Attribute Name="ProjectSign" AttributeDataType="xs:string" />
<Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
<Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
<InternalElement ID="b27045c4-9cb3-4b8d-916b-85f8100d1602" Name="Ungrouped devices">
<InternalElement ID="3f770698-940d-49c2-9f77-06fc458e1340" Name="ET 200SP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.ET200SP</Value>
</Attribute>
<InternalElement ID="6f52fbab-a221-4d54-9368-84c392ca7fec" Name="Rack_0">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
...
<InternalElement ID="f7445c0b-1c52-4a84-915f-2c8bee13af70" Name="BA 2xRJ45">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>BA 2xRJ45</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>127</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6AR00-0AA0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V0.0</Value>
</Attribute>
<InternalElement ID="40f8bbce-35d3-4d65-907a-bece3e0144e0" Name="PROFINET interface">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
```

```
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="8fb775eb-96c6-48d6-af8a-96ba72418830" Name="IE1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<InternalElement ID="f28a3d93-d821-4556-9df1-a45f0e4ff6a6" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="ad6a0faa-3b70-4528-8c54-8183018b6714" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
...
```

Note

CAx shall perform export and import of AR APC version in AML file in accordance to the installed version of TIA Portal.

Import sub modules

You can import sub modules from an AML files, which is generated from above export.

Note

- The export hierarchy change behavior shall be applicable only in versions V15.1 onwards
- The hierarchy in AML file shall not influence/affect the TIA Portal internal hierarchy after Import.
- The AML files created using older TIA Portal versions shall also be imported without any failure.
- This hierarchy change/transformation behavior is applicable for both built-in and non built-in sub modules.

Multiple sub modules under same Interface

There are some scenarios where multiple sub modules shall exist under a same interface. For ex: IO Device : IM 155-6 PN/3 HF 6ES7 155-6AU30-0CN0/V4.2. This head module has two non built-in Bus Adapters under a same interface. In such case, it shall be possible to export mentioned Bus Adapters from TIA Portal hierarchy to required AML file hierarchy. In this example from TIA Portal hierarchy, 'PROFINET interface' has two Bus Adapters, three ports and one node. Here, Port_1 and Port_2 logically belongs to BA 2xRJ45 and Port_3 logically belongs to BA 2xRJ45_1 though all the three ports are aggregated under one interface.

During Export:

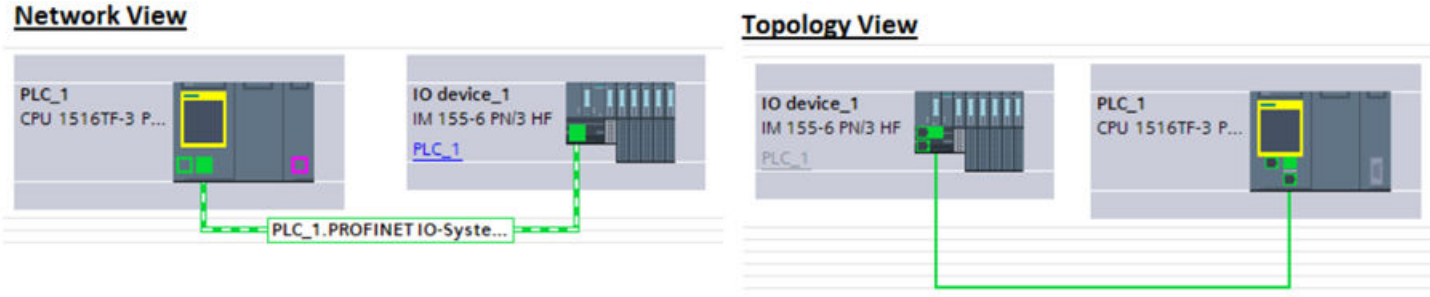
- AML file should be exported with Label and Type attribute supported for secondary interface
- Only first sub module shall get 'original' interface along with its connection relevant information. Here, BA 2xRJ45 gets original interface along with node 'IE1', 'Port_1' and 'Port_2'.
- Rest of the sub modules shall get a 'duplicate' interface with ports which logically belongs to the sub module. Here, BA 2xRJ45_1 shall get a 'duplicate' interface and Port_3.
- If the head module is connected to a subnet/Iosystem, the relevant link information(like ExternalInterface links) shall be exported only as part of first sub module (ExternalInterface link related to Subnet under 'Node' and ExternalInterface link related to IoSystem under 'Interface') .
- The link information pertaining to topology connection, shall be part of respective 'Port'.

During import:

- It shall be possible to import multiple sub modules from an AML file which is generated out of above mentioned export.
- It shall be possible that EPLAN generated AML file shall have node information inside secondary interface

- Node details, which is duplicate of primary interface node are handled as mentioned below:
  - Node Attributes: Overwrites node attribute details which set during Primary interface processing
  - Subnet Connection: Silently skipped if it is already connected otherwise connection shall be established
- If the AML file contains IoSystem connection details on secondary interface, then
  - Connection shall be skipped if already connected and user shall be notified with proper error message in InfoTab.
  - Connection shall be established if it is not connected.

The following configuration shows IO device configuration with master-slave and topology connections.



The below example shows a partial AML file that shall be generated during export for the above configuration:

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile FileName="MultipleBA_01.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
<WriterName>Totally Integrated Automation Portal</WriterName>
<WriterID>1d4fcebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
<WriterVendor>Siemens AG</WriterVendor>
<WriterVendorURL>www.siemens.com</WriterVendorURL>
<WriterVersion>15</WriterVersion>
<WriterRelease>1501.0000.0.0</WriterRelease>
<LastWritingDateTime>2018-05-17T09:36:46.9230179Z</LastWritingDateTime>
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="e005c094-1b0a-42c4-92a0-67c981508c1a" Name="Project45">
<Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
<Attribute Name="ProjectSign" AttributeDataType="xs:string" />
<Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
<Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
<InternalElement ID="2782e61d-8c27-46cb-93ea-6b804157ae60" Name="PN/IE_1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<ExternalInterface ID="2d901881-a2bf-4fe7-915f-b2542b346988" Name="LogicalEndPoint_Subnet"
RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Subnet" />
...
<InternalElement ID="dc5cf410-2516-4b0b-ad1a-c43117d8c9b3" Name="BA 2xRJ45">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>BA 2xRJ45</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>127</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6AR00-0AA0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V0.0</Value>
</Attribute>
<InternalElement ID="f04874a8-2d35-47c4-93ae-d6fdc2668479" Name="PROFINET interface">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
```

```
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="81a7d9df-99b8-4eca-8e72-404b22bd05e7"
Name="LogicalEndPoint_Interface" RefBaseClassPath="CommunicationInterfaceClassLib/
LogicalEndPoint" />
<InternalElement ID="83eb7d69-8cd5-4217-a07a-0c656d215ec7" Name="IE1">
...
```

Devices with sub module and integrated port at Interface level

There are some device configuration where a module can support an integrated port as well as extended ports(For ex: via sub module i.e., Bus Adapter). For ex: ET200SP CPU: CPU 1510SP-1 PN/6ES7 510-1DJ00-0AB0/V1.8. This CPU module supports three ports (two ports via Bus Adapter and one integrated port) under a same interface.

In this example from TIA Portal hierarchy, 'PROFINET interface' has Bus adapter, three ports and one node. Here, Port_1 and Port_2 logically belongs to BA 2xRJ45 and Port_3 logically belongs to PROFINET interface though all the three ports are aggregated under one interface.

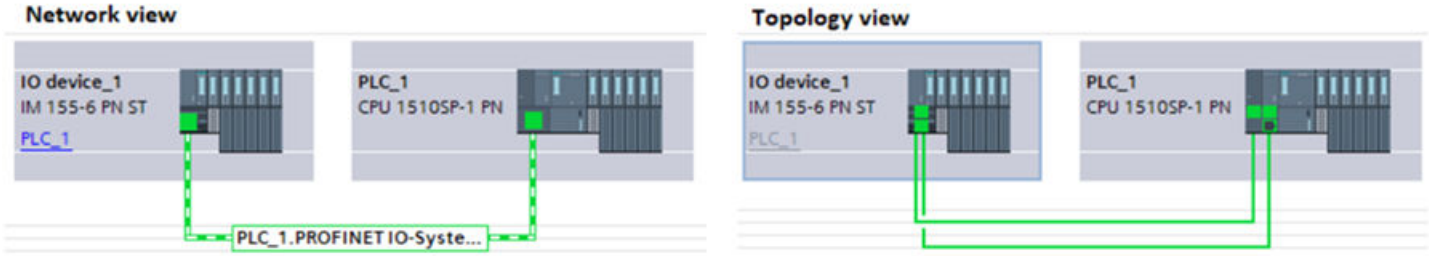
During export:

- The sub module shall get 'original' interface along with its connection relevant information. Here, BA 2xRJ45 gets original interface along with 'IE1', 'Port_1' and 'Port_2'. And head module shall have the 'duplicate' interface (having Type-'Ethernet') with only integrated port 'Port_3' in it.
- In case sub module (BA 2xRJ45) is not plugged, then PROFINET interface at head module level shall be considered as 'original' interface and shall have 'IE1' and 'Port_3'.
- If the head module is connected to a Subnet/IoSystem, then relevant link information (like ExternalInterface links) shall be exported only as part of 'original' interface (ExternalInterface link related to Subnet under 'Node' and ExternalInterface link related to IoSystem under 'Interface').
- The link information pertaining to topology connection, shall be part of respective 'Port'.

During import:

- It shall be possible to import modules with interface having sub module and integrated port under it from an AML file which is generated out of the above-mentioned export.
- Handling redundant information (Node, IoSystem and links) under 'duplicate' interface shall be handled in a same way as 'Multiple sub modules under same interface' scenario.

The following configuration shows modules with master-slave and topology connections.


Network view
IO device_1
IM 155-6 PN ST
PLC_1
PLC_1
CPU 1510SP-1 PN
PLC_1.PROFINET IO-Syste...
Topology view
IO device_1
IM 155-6 PN ST
PLC_1
PLC_1
CPU 1510SP-1 PN

The AML file that shall be generated during export for the above configuration is depicted below:

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile FileName="Project_BusAdapter_Demo.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
...
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.2.0" />
</AdditionalInformation>
<InternalElement ID="12b43940-cea6-476a-886c-11ebaa518256" Name="BA 2xRJ45">
...
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6AR00-0AA0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V0.0</Value>
</Attribute>
<InternalElement ID="8f8a4cdb-8f4b-4e72-8c92-b8fa1cc3bf70" Name="PROFINET interface_1">
...
<InternalElement ID="7a9938c3-ccc0-4d28-be35-333a343f3613" Name="E1">
...
<ExternalInterface ID="3d9f7f55-723f-4c5d-a2e7-ffe1ec3b9167" Name="LogicalEndPoint_Node"
RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationEthernetRoleClassLib/
NodeEthernet" /</InternalElement>
<InternalElement ID="3c4085cb-7565-4673-8de1-c5624c4c08dc" Name="PROFINET IO-System">
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>100</Value>
</Attribute>
<ExternalInterface ID="7fb6129f-95c4-4d2c-aab5-702937198e80"
Name="LogicalEndPoint_IoSystem" RefBaseClassPath="CommunicationInterfaceClassLib/
LogicalEndPoint" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
IoSystem" />
</InternalElement>
<InternalElement ID="1f1d3e8d-55da-4355-b87e-7feb58d86143" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1R</Value>
</Attribute>
...
<ExternalInterface ID="850e3f32-985f-4432-b627-26e2775a69cc"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="11731ed5-7922-41c9-b179-a5ae029cc10d" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2R</Value>
</Attribute>
...
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" /> </InternalElement>
```

```
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="7be36254-5ed6-4a6f-9e7b-90be8b35e595" Name="PROFINET interface_1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
...
<InternalElement ID="61ac6187-8a5d-4f98-ae91-b809c0a3a15d" Name="Port_3">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P3</Value>
</Attribute>
...
<ExternalInterface ID="425f5a5d-84e2-40c6-928f-e1aab73a8b86"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
```

Pruned AML

Pruning is an act of optimizing the content by removing certain things which are not necessary to be provided. For information on Pruned AML, (Refer Pruned AML (Page 1431)).

There might be some scenarios where sub module configuration hierarchy is not same in TIA Portal and CAD tools (like EPALN) due to pruned sub modules. In such scenarios, TIA Portal shall support import of both pruned and unpruned AML files.

Note

- TIA Portal shall always export unpruned AML file.
- TIA Portal shall always import both pruned and unpruned AML file

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.17 Export/Import AML file in UMAC environment

Introduction

With TIA Portal V17, it shall be possible to perform CAx export and import operations on a UMAC protected project.

CAx operations on protected project is based on the below mentioned functions rights:

- Project with read and write access
- Modify project via Openness API

Export

CAx export operation is not restricted. So, it shall be possible to perform export operations on protected project irrespective of the above mentioned users function rights.

Import

CAx import operation is restricted. When a user has above mentioned access rights then import shall be successful, otherwise missing function rights error message shall be displayed in TIA Portal user interface or Function rights not found exception shall be thrown for CAx API.

User function rights and CAx operation

Below table depicts the users function rights and their allowed CAx operations:

Users	Function Rights		CAx Operations	
	Project with read and write access	Modify project via Openness API	Export	Import
Project Administrator	X	X	X	X
Project read-only user	-	X	X	-
User without Openness access	X	-	X	-
Read-only user without Openness access	-	-	X	-

Note

- From TIA Portal V17 onwards, it shall be possible to perform CAx Export operation on read-only project.

To perform Export and Import on UMAC protected project using CAx API, project needs to be opened in UMAC protected environment. see Opening a Project (Page 128)via API for opening project in UMAC protected environment.

See also

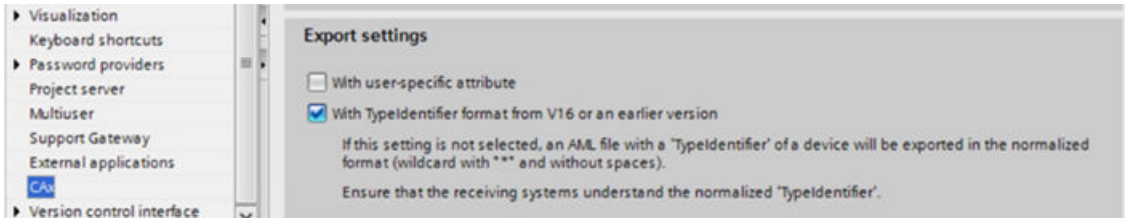
Opening a project (Page 128)

6.5.18 Export/Import AML file with normalized type identifier

Introduction

In TIA Portal, there are different notations for order numbers inside type identifiers. In order to allow import and export of these different notations of order numbers to integrate with other tools, Tolerant handling of Order Numbers in Automation AML files is now possible.

In order to export TypeIdentifier format of TIA Portal V16 and below, a new setting has been added:



If this checkbox "With TypeIdentifier format from V16 and below" is checked, TypeIdentifier value in AML will be exported in old format, e.g. OrderNumber:6ES7 516-3AN00-0AB0.

If this checkbox is unchecked, TypeIdentifier value in AML will be exported with Order number's wildcard characters denoted by "*" and blanks will be removed, e.g. OrderNumber:6ES7590-1***0-0AA0.

During the import arbitrary wild card characters and blanks are accepted as part of the order number of a TypeIdentifier.

6.5.19 Import CAx data without logical address

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)

Application

In TIA Portal, you can use CAx import to configure connection between channel and tag without having the start address of an I/O module and/or logical address of tag specified in the AML file.

The following AML file example depicts the XML that shall be generated without start address and logical address attributes.

```
<?xml version="1.0" encoding="utf-8"?><CAEXFile FileName="TagsExport.aml"
SchemaVersion="2.15" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="fff25423-fe9e-4334-9331-4cec118e06f7" Name="Project1">
...
<InternalElement ID="59c0b48b-aa6c-45c3-8dc8-bba5367bd4fb" Name="S7300/ET200M station_1">
...
<InternalElement ID="1b7b2b24-243b-4348-831e-bc46bc35957f" Name="Rail_0">
...
<InternalElement ID="974ca791-ad8d-482b-be80-2cf4e8dcedaf" Name="PLC_1">
...
<InternalElement ID="9564bcc2-8ea0-4be7-a950-5c55b34e474a" Name="Default tag table">
<ExternalInterface ID="7fd969e6-c2c9-45a8-b573-68833df327f5" Name="Tag_1"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
<Attribute Name="DataType" AttributeDataType="xs:string">
<Value>Bool</Value>
</Attribute>
</ExternalInterface>
<ExternalInterface ID="33899862-86c1-4171-832a-1136b6e59b9d" Name="Tag_2"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
<Attribute Name="DataType" AttributeDataType="xs:string">
<Value>Byte</Value>
</Attribute>
</ExternalInterface>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
TagTable" />
</InternalElement>
...
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>8</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<Attribute Name="Address">
<RefSemantic CorrespondingAttributePath="OrderedListType" />
<Attribute Name="1">
<Attribute Name="StartAddress" AttributeDataType="xs:int">
<Value>832</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>128</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
<Attribute Name="2">
<Attribute Name="StartAddress" AttributeDataType="xs:int">
<Value>832</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>128</Value>
</Attribute>
```

```
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Output</Value>
</Attribute>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="29e0bb63-0050-46e3-968a-fcecf4eb050a" Name="DI 16x24VDC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>DI16 x 24VDC</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>4</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 321-1BH02-0AA0</Value>
</Attribute>
<Attribute Name="Address">
<RefSemantic CorrespondingAttributePath="OrderedListType" />
<Attribute Name="1">
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>16</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
</Attribute>
<ExternalInterface ID="175dc9c9-f9a3-4b10-b43e-68dfc14811fc" Name="Channel_DI_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Digital</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
</ExternalInterface>
<ExternalInterface ID="23e99053-906c-4548-9bd2-e975cacf01b2" Name="Channel_DI_1"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Digital</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
```

```
<Value>Input</Value>
</Attribute>
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
</ExternalInterface>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
<InternalLink Name="Link To Tag_1" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_0" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_1" />
<InternalLink Name="Link To Tag_2" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_0" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
<InternalLink Name="Link To Tag_3" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_1" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
<InternalLink Name="Link To Tag_4" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_2" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
AutomationProject" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

Tag with Bool datatype (%I0.0)

For the above AML file example, upon import the Logical Address for tag is calculated by using the following algorithm:

Logical Address = ChannelIoType + ByteAddress + BitAddress

Name	Description
ChannelIoType	Input (I) or Ouput (Q)
ByteAddress	ByteAddress shall be calculated as specified below : StartAddress of the I/O Module * 8 + BitOffsetAddress of the I/O Module + (ChannelNumber * ChannelLength) / 8
BitAddress	BitAddress shall be calculated as specified below: (BitOffsetAddress of the I/O Module + (ChannelNumber * ChannelLength)) % 8

Note

- In cases where a tag spans across more than one modules: The algorithm above gives multiple byte addresses based on the start address of modules and an array of bit addresses per byte address, corresponding to each channel number. Then the algorithm shall select the lowest byte address and the lowest among the bit address array corresponding to this byte, for calculating the logical address of the tag. Once the logical address is assigned to the tag, the spanning is automatically taken care by the TIA Portal during import
- In TIA Portal, only few device configuration (for example ASI modules) supports Bit Offset address attribute. For modules which does not support BitOffset address attribute, default value "0" will be considered for above mentioned calculation.

Following are the list of tag data types which supports Bit address in TIA Portal:

Tag data types	Bit address value
Bool	0 to 7
LReal	Highest and lowest bit number values is 0 in TIA Portal.
LWord	
LInt	
ULInt	
LTime	
LDT	
LTime_Of_Day	

During CAx import, If a boolean tag is configured to a channel type, then logical address calculation includes bit offset address. The logical address along with bit offset shall be updated as the tag's logical address in TIA Portal UI.

Tag with other datatypes which supports Bit Offset Address

In case of tag channel configured between two different datatypes where a channel of type bool is mapped to a tag of type LDT which spans across multiple channels, then logical address calculation includes bit offset address for calculating logical address for Tag of "LDT" datatype and the tag logical address will be updated in TIA Portal UI with an error if the bit address value is other than "0" and tag channel configuration shall not happen.

You must make sure that tag-channel configuration is happening between similar datatypes.

Tag with with other datatypes (%IB0)

Logical Address = ChannelIoType + TagDataType + ByteAddress

Name	Description
ChannelIoType	Input (I) or Ouput (Q)
TagDataType	TagDataType is an abbreviation of the tag type. Example: W for word and B for Byte
ByteAddress	ByteAddress shall be calculated as specified below : StartAddress of the I/O Module + (ChannelNumber * ChannelLength) / 8

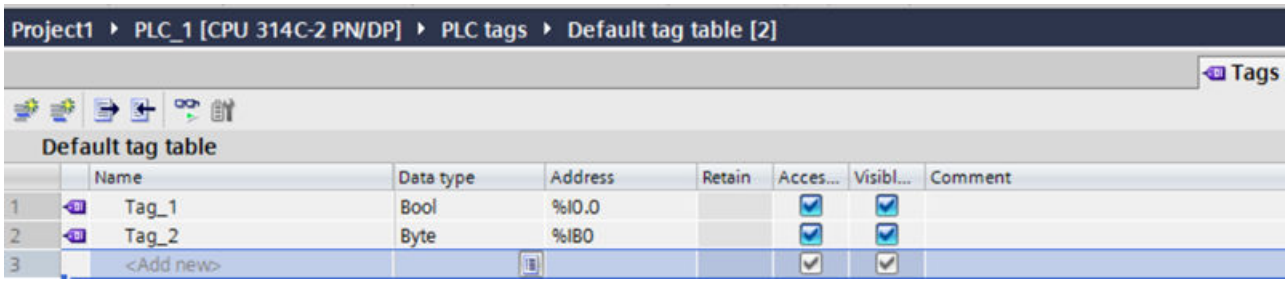
The algorithm explained above is used to calculate the Logical Address of a tag accurately in case of

- Length of the datatype specified in the tag should be equal to the length of the channel it is mapped to.
- For example, If a tag of datatype "Byte" is mapped to an Analog channel which is of length 2Bytes: upon import of AMl file which doesnot have Logical Address of the tag specified; The tag in TIA portal shall always be mapped to the first byte of the channel irrespective of which byte it was originally mapped to.

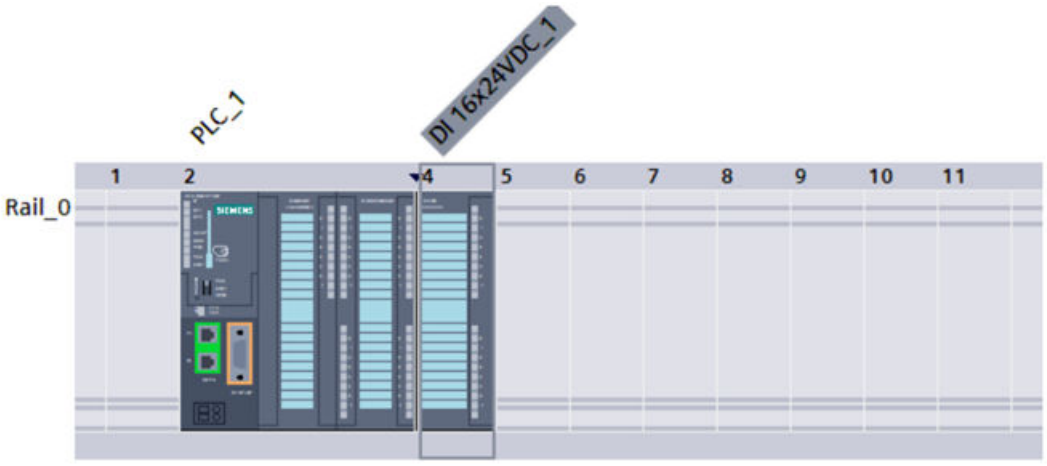
Note

- In cases where a tag spans across more than one modules: The algorithm above gives multiple byte addresses based on the Start Address of the modules. Then the algorithm shall select the lowest byte address for calculating the logical address of the tag. Once the logical address having the lowest byte address is assigned to the tag, the spanning is automatically taken care by the portal during import. If the StartAddress attribute for the I/O Module is not provided in the AML file, the default value is assigned by TIA portal and same shall be used for the above calculation.
- If the StartAddress attribute for the I/O Module is not provided in the AML file, the default value is assigned by TIA portal and same shall be used for the above calculation.

Upon successful completion of import the following tag configuration shall be created in TIA Portal for the example above.



The channels shall be configured to the tags as depicted below.



See also

Connecting to the TIA Portal (Page 82)

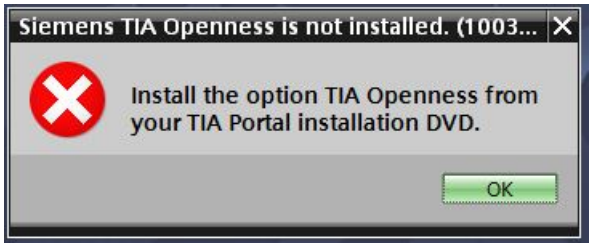
Opening a project (Page 128)

6.5.20 Exceptions during import and export of CAx data

Exception due to non-availability of TIA Openness

CAx implementation is based on TIA Openness Public API's. Openness Public API's are available only when user has installed Openness optional pack during TIA Portal installation. Hence, there is a need to check whether Openness is available before performing any CAx related functionalities. (Refer Installing TIA Portal Openness (Page 38))

Whenever user triggers a CAx Export or CAx Import actions from TIA Portal UI, a check is performed to see availability of TIA Openness in the system. If TIA Openness is not found to be installed, user will be displayed a TIA Portal message dialog with following error message dialog.



6.5.21 Round trip exchange of devices and modules

Requirement

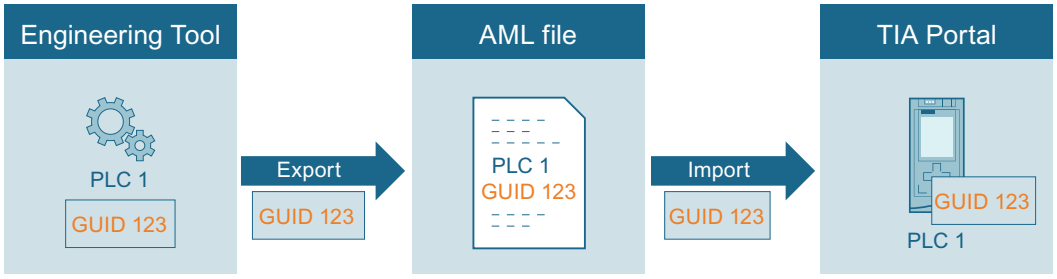
- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

You can exchange configuration data between the TIA Portal and other engineering tools, e. g. an electrical planning tool like EPLAN or the TIA selection tool. For the identification of the imported and exported devices, a global unique identifier, the AML GUID, is used.

During the round trips, the AML GUID is kept stable for physical assets like devices and device items which are not built-in e.g. CPUs or modules, but not for virtual assets like tags, channels, ...

During the first export from the TIA Portal, the AML GUID for a device or a no built-in device item is randomly generated, but kept stable afterwards.

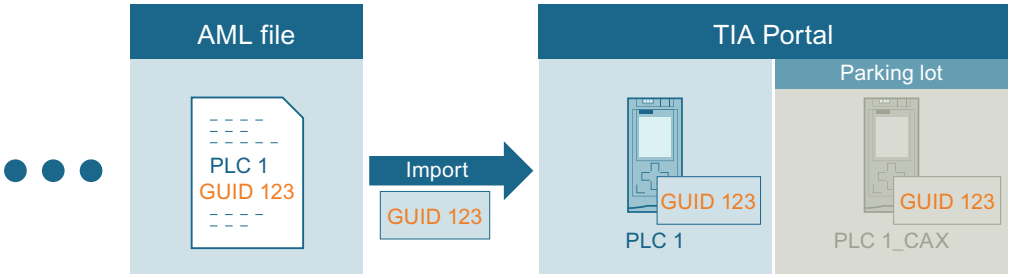


If you export a device from an engineering tool into an empty TIA Portal project, the AML GUID is added to the comment of the hardware object. If in the TIA Portal at "Tools > Settings > CAx > Import settings" the correspondig setting is enabled the AML GUID is added in the current editing language. The round trip process supports only one editing language to store the AML GUIDs. When importing or exporting data, always use the editing language with which you started the round trip.

For all following imports or exports, the AML GUID stays the same for this hardware object. Changes to the hardware object are resumed.

Within a TIA Portal project object names have to be unique. The import of a device or a device item into a TIA Portal project where a certain object with the same name already exists would lead to a naming conflict. During the import you have the possibility to move the objects with naming conflicts to the user defined parking lot. The name of the imported Object will be extended with "_CAX".

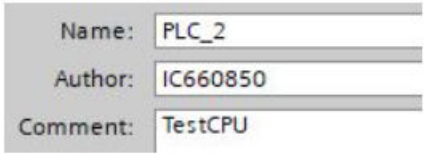
During CAx export of the GUI



To support AML roundtrip for previous project version, during project upgrade AML GUIDs are now stored in "CustomIdentity" (App ID) instead of "Comment" part of device/deviceitem. The AML GUIDs of device/deviceItem are expected to be unique and could be stored in any editing language. So during project upgrade, all the comment editing languages are considered to get the unique AML GUID. As the AML GUID of device/deviceItem is unique, but when a new GUID with matching [AR_APC:ID:*] regex is added to comment in any editing language then the first GUID which is picked from the editing languages is considered as the AML GUID for that device/deviceItem.

Once the AML GUID's in the comment are moved to CustomIdentity, the device/deviceItem comment is updated by removing the [AR_APC:ID:*] as shown in the image below

If the comment contains multiple [AR_APC:ID:*] section then the first GUID which matches the pattern is set to CustomIdentity repository and the same is removed from comment. Rest of the text is considered as comments.



Note

Copying an imported device

If you copy a device or a device item possessing an AML GUID you have to delete the AML GUID in the comment of the copied object. Otherwise, devices or device items with identical AML GUID exists in your project and lead to an invalid AML file.

Import settings

1. Define the parking lot folder name under "Options > Settings > CAx > Settings for conflict resolution".
   The parking lot folder is used to store objects with naming conflicts.
2. Activate "Options > Settings > CAx > Import settings > Save GUIDs during import".

During Import:

- The GUIDs of a physical asset shall be stored as part of its CustomIdentity section
- While importing an AML file, the GUIDs are stored as part of CustomIdentity

Note

Valid AML GUID

If you edit an AML GUID before the import, the AML GUID becomes invalid and then CAx import operation shall be aborted and corresponding information shall be logged.

Note

Exceeding length of a comment

If the appending of the AML GUID the comment exceeds its maximum limit of 500 characters, the user comment value will be trimmed to 500 characters. A corresponding information will be logged.

Export

During export:

- GUID to be exported shall be fetched from CustomIdentity for the key AR_APC:ID.
- If the GUID is not available as part of CustomIdentity then the Multi user GUID shall be fetched and exported as the AML GUID
- If above 2 steps fail to provide the GUID of a physical asset then a new random GUID shall be generated by the export process and shall be treated as the AML GUID for physical assets.

AML structure

The generated ID is exported to AML file as depicted in the following code snippet:

```
<InternalElement ID="23aeefd0-ce05-4116-a644-e33d43901eaf"
Name="PLC_1"
```

6.5.22 Export/Import topology

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

In TIA Portal, you can export the devices with its topology information to an AML file. While importing to an empty TIA Portal project, the imported device items retains the topology information.

<InteralLink> element gives link details of port to port interconnection between the device items. It appears under the common parent of the connected devices, and contains unique tag names.

Attributes of a "InternalLink" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	The tag names are formatted as "Link to Port_n" (where n varies from 1 to the number of port to port links).
RefPartnerSideA	Mandatory	Denotes the port which are linked. Formatted as UniqueIDOfPort:CommunicationPortInterface
RefPartnerSideB	Mandatory	Denotes the port which are linked. Formatted as UniqueIDOfPort:CommunicationPortInterface

Example: Topology view

AML structure

The following figures shows a partial element structure of the exported AML file. It contains two unique ID for the ports in PLCs.

```
...
 <InternalElement ID="e1966b52-b8b3-47b4-8866-a754ebb77648" Name="Port_1">
   <Attribute Name="Label" AttributeDataType="xs:string">
```

```
...
 <InternalElement ID="75f31daf-575f-48a2-ab35-8f07a376eb1b" Name="Port_1">
   <Attribute Name="Label" AttributeDataType="xs:string">
```

The <InteralLink> element contains three mandatory attributes.

```
<InternalLink Name="Link to Port_1"
  RefPartnerSideA="e1966b52-b8b3-47b4-8866-a754ebb77648:CommunicationPortInterface"
  RefPartnerSideB="75f31daf-575f-48a2-ab35-8f07a376eb1b:CommunicationPortInterface" />
```

6.5.23 Import of device with Library references

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

You can import device (station)/device item (module, sub-modules) in AML file into TIA Portal using Library references.

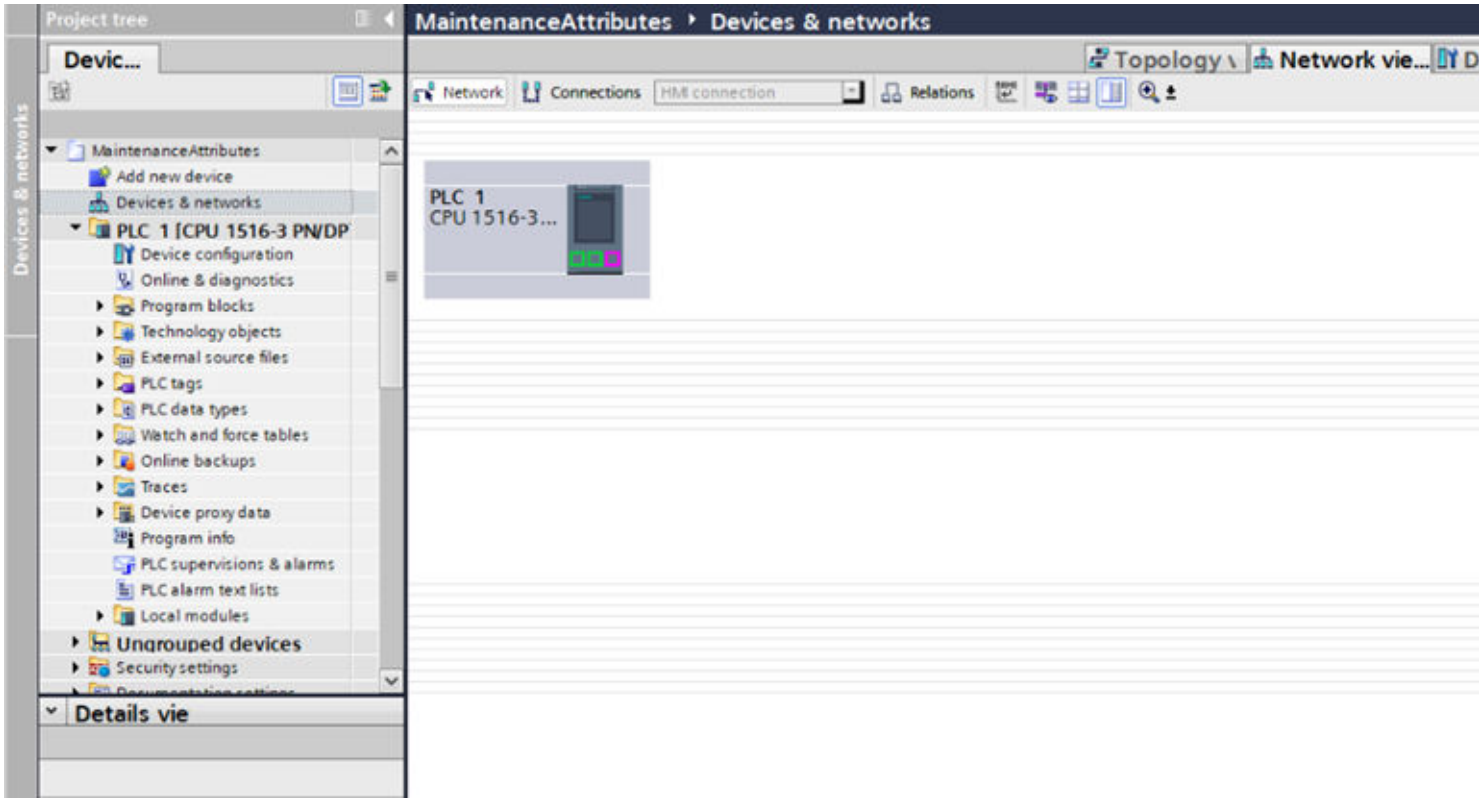
AML Structure

The following AML structure describes the XML file that shall be used during the import operations into TIA Portal project.

```
...
<InternalElement ID="bed34f88-7a3f-4e37-a32f-df1a6dcb954a" Name="S71500/ET200MP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.S71500</Value>
<Attribute Name="TemplateIdentifier" AttributeDataType="xs:string">
<Value>GlobalLib://StationPlcLibrary/Master copies/DeviceFolder/S71500ET200MP station_1</
Value>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" />
...
<InternalElement ID="5082f437-4198-4dcb-b794-35b6b9fcd104" Name="PLC_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 214-1BE30-0XB0</Value>
<Attribute Name="TemplateIdentifier" AttributeDataType="xs:string">
<Value>GlobalLib://StationPlcLibrary/Master copies/PLC_1</Value>
</Attribute>
</Attribute>
</InternalElement>
...
</InternalElement>
...
```

Example: Imported configuration

The AML snippet above corresponds to station and PLC in TIA Portal is depicted below:



The feature is delivered considering the following points:

- The Scope of this feature restricts the usage of libraries only to Global Libraries.
- In case a device/device item comes with TemplateIdentifier attribute, this has more priority over Type Identifier.
- The Global Libraries used in the import operation are assumed to be loaded prior to the onset of import, failing which an appropriate error shall be displayed to the user.
- Amongst all the System Folders found inside a library only the 'Master Copies' folder shall be considered for import operation.
- In order to support multilingual character of TIA Portal, TemplateIdentifier can have Master Copy path in all supported TIA languages but should be preceded with "GlobalLib://". And the user of the Portal has to make sure that the TIA UI language and the language in the AML file are one and the same.
- For an error free import operations the device/device item master copy in the library and the device/device item in the AML file shouldn't have conflicting items. Conflicts here can result in loss of data as well.
- If the device/device item in the AML file has generic Type ID or no Type ID value at all, then the regular workflow of Type ID substitution shall operate as usual. But the device/device item shall be created from TemplateIdentifier and after the creation the Type ID shall not be cross verified with the substituted Type ID.

- In the AML file the TemplateIdentifier sub-attribute under TypeIdentifier attribute can feature on any Pluggable item and only on pluggable items. Any other entity in the AML file (e.g. Channel/Tag) if it has a Template reference, shall be treated invalid and appropriate message to be displayed to the user.
- The character '/' in the TemplateIdentifier path shall be considered as a delimiter for folder segregation. In case a master copy object has a name having character '/', this might result in CAx not able to find the master copy object.
- The import merge use cases with respect to device configuration having TemplateIdentifier should work in same way as that of device configuration having TypeIdentifier
- Existing merge use cases with AML file having only TypeIdentifier should also behave as usual.
- In case of a conflict where the same device is present in both TIA and the AML file (with a library reference to it), the Device in library if in this case has the PLC inside, the merge operation shall only be able to rename the Station but not the PLC.
- There is a limitation in setting start address for input output modules when they are imported from mastercopy. Once a start address is already assigned to a module, then it is not possible to set the start address when it is created from TemplateIdentifier during import. The important thing to note here is that users shall add the module into the mastercopy (while preparing global libraries) before an address is assigned to it , that means the module should not be connected to its master controller before it is put into the master copy.

Note

- A V16/V17 AML file with device/device item having 'TemplateIdentifier' should be imported successful in TIA Portal V17
- A V16/V17 AML file with device/device item having 'TemplateReference' should not be imported successfully in TIA Portal V17
- A V16 AML file with device/device item having 'TemplateReference' should be successful in V16. This behavior for V16 TIAP shall not be changed.

- Prior to TIA Portal V17, device configuration which does not support module level(especially main modules like Head module/CPU) library drag and drop support in TIA Portal could not be supported in CAx library workflows.
  However, this kind of configuration supports complete device level library drag and drop where user shall include pre-configured rack and main module as part of device master copy itself and put it in to library.

With this support from TIA Portal V17, CAx shall also extend to Import the whole device master copy from library which shall have preconfigured rack and main module in it

Below are some of the expected behavior for importing preloaded device master copy using AML file.

- In TIA Portal V17, any AML file with device configuration having device level library reference (with rack and main module) should import successfully by creating device from library by retaining rack and main module also from device master copy.
  - Provided, Library reference is valid
  - Provided, TypeIdentifier of rack and main module inside library is inline with respective AML TypeIdentifier
- In TIA Portal V17, any AML file with device configuration having device level library reference (with rack and main module) should import with error by still creating device from library with retaining rack and main module also from device master copy
  - When Library reference is valid
  - When TypeIdentifier of rack or main module inside library is not matching with respective AML TypeIdentifier
- In TIA Portal V17, any AML file with device configuration having device level library reference (with rack and main module) should import without a warning by still creating device from library with retaining rack and main module also from device master copy, when
  - When Library reference is valid
  - When FW of rack or main module inside device master copy is not matching with respective AML FW
- In TIA Portal V17, any AML file with device configuration having device level library reference (with rack and main module) should import without a warning by still creating device from library with retaining rack and main module also from device master copy.
  - When Library reference is valid
  - When FW of rack or main module is missing in AML

Recommendation for library reference usage

In general, CAx shall support multiple ways of handling library references during CAx Import. The user shall use any of the allowed way to achieve successful import of an AML file. For example:

1. If TIA Portal supports library workflow (drag and drop support) for main module(like Head module/CPU) in the device configuration as an "independent" master copy, then CAx user shall configure the AML file with device/device items as having "independent" library reference. Here, expectation is that main module exist as an independent master copy in library.
2. If TIA Portal does not support library workflow (drag and drop support) for main module(like Head module/CPU) in the device configuration as an "independent" master copy, then CAx user shall configure the AML file with device having template identifier. Here, expectation is that main module exist as part of complete device master copy in library.

Note

Every device configuration has different behavior wrt library workflows (drag and drop) in TIA Portal. Some device families supports library workflows at granular level (device item level) , some at only complete device level and some even support both. Hence, CAx user shall configure AML files with library references by keeping these TIA Portal aspects.

Whitelist and Blacklist

- If a device/deviceitem is mentioned in the whitelist/blacklist, it means
  - either the device/deviceitem has been verified as an 'independent' master copy without any sub device items.
  - or the device configuration has been verified as a 'complete' device master copy with rack and main module in it.
- It is possible that a device item can be used in multiple device configurations. However, if it is mentioned in the whitelist does not mean that library import for that device item works in all the possible configuration.
  - For ex: If a device item - x is mentioned with firmware version - y, then
    - It might not work for different firmware version
    - It might not work when plugged inside different module/station

Below table shows list of device/deviceitems that supports import using library references.

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 212-1HD30-0XB0	
OrderNumber:6ES7 510-1DJ00-0AB0	
OrderNumber:6ES7 221-3BD30-0XB0	
OrderNumber:6ES7 511-1CK00-0AB0	V2.2
OrderNumber:6ES7 516-3AN01-0AB0	V1.8
OrderNumber:6ES7 518-4AP00-0AB0	V1.0
OrderNumber:6GK7 542-5DX00-0XE0	V1.8
OrderNumber:6ES7 611-4SB00-0YB7	V1.8
OrderNumber:6ES7 315-2FJ14-0AB0	V1.8
OrderNumber:6ES7 154-8AB01-0AB0	V1.0
OrderNumber:6ES7 141-4BF00-0AA0	V4.6
OrderNumber:6ES7 137-6BD00-0BA0	V3.1
OrderNumber:6ES7 147-4JD00-0AB0	V3.2
OrderNumber:6ES7 154-4AB10-0AB0	
OrderNumber:3RK7 137-6SA00-0BC1	V2.2
OrderNumber:6ES7 518-4FP00-0AB0	V1.0
OrderNumber:6GK7 543-1AX00-0XE0	V7.0/Cu
OrderNumber:6ES7 516-2GN00-0AB0	V1.1
OrderNumber:6ES7 148-4CA00-0AA0	V2.5
OrderNumber:6ES7 148-4CA60-0AA0	V2.1
OrderNumber:6ES7 154-8FX00-0AB0	V2.1
OrderNumber:6ES7 148-4EA00-0AA0	
OrderNumber:6ES7 151-7AA20-0AB0	
OrderNumber:6ES7 138-4CA50-0AB0	V3.2
OrderNumber:6ES7 151-8AB00-0AB0	
OrderNumber:6ES7 138-4CB11-0AB0	V2.6
OrderNumber:6ES7 151-7AA21-0AB0	
OrderNumber:6ES7 138-4CA60-0AB0	V2.7
OrderNumber:6ES7 151-7FA21-0AB0	
OrderNumber:6ES7 138-4CA01-0AA0	V3.3
OrderNumber:6GT2002-0HD00	
OrderNumber:6ES7 142-4BD00-0AA0	V3.3
OrderNumber:6SL3 235-0TE21-1RB0	
OrderNumber:6SL3 235-0TE21-1SB0	V1.0
OrderNumber:6SL3 514-1KE13-5AE0	
OrderNumber:6ES7 148-4EB00-0AA0	V3.1
OrderNumber:6ES7 143-4BF00-0AA0	V3.1
OrderNumber:6ES7 134-6JD00-0CA1	V4.7.6
OrderNumber:6ES7 193-6PA00-0AA0	
OrderNumber:6ES7 512-1SK00-0AB0	
OrderNumber:3RK1 308-0AB00-0CP0	V2.0
OrderNumber:6ES7 512-1DK00-0AB0	V1.1

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 137-6BD00-0BA0	V1.8
OrderNumber:6ES7 135-6HD00-0BA1	V1.1
OrderNumber:6ES7 137-6AA00-0BA0	V1.8
OrderNumber:6ES7 510-1SJ01-0AB0	V2.1
OrderNumber:6ES7 138-6AA00-0BA0	V1.1
OrderNumber:3RK1 308-0BE00-0CP0	V1.0
OrderNumber:6ES7 135-6HB00-0DA1	V2.1
OrderNumber:6ES7 134-6FF00-0AA1	V1.2
OrderNumber:3RK1 308-0AE00-0CP0	V1.1
OrderNumber:6ES7 390-1***0-0AA0	V2.0
OrderNumber:6ES7 307-1BA00-0AA0	V1.0
OrderNumber:6ES7 318-3FL01-0AB0	V1.1
OrderNumber:6ES7 338-7XF00-0AB0	
OrderNumber:6ES7 307-1KA01-0AA0	
OrderNumber:6ES7 317-2FK13-0AB0	V3.2
OrderNumber:6GT2002-0GA10	
OrderNumber:6ES7 307-1EA00-0AA0	
OrderNumber:6ES7 317-6FF04-0AB0	V2.6
OrderNumber:6ES7 323-1BH01-0AA0	
OrderNumber:6ES7 307-1EA80-0AA0	
OrderNumber:6ES7 315-2FJ14-0AB0	V3.3
OrderNumber:6ES7 350-1AH03-0AE0	
OrderNumber:6ES7 307-1EA01-0AA0	
OrderNumber:6ES7 318-3EL00-0AB0	V3.2
OrderNumber:6ES7 321-7EH00-0AB0	
OrderNumber:6ES7 307-1KA02-0AA0	
OrderNumber:6ES7 317-2EK13-0AB0	V2.8
OrderNumber:6ES7 323-1BL00-0AA0	
OrderNumber:6ES7 314-6BG03-0AB0	
OrderNumber:6ES7 340-1AH02-0AE0	V2.6
OrderNumber:6ES7 313-6BG04-0AB0	

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 360-3AA01-0AA0	V2.6
OrderNumber:6ES7 314-1AG13-0AB0	V1.0
OrderNumber:6ES7 321-7TH00-0AB0	V3.3
OrderNumber:3RK1 335-0AS01-0AA0	
OrderNumber:3RK1 305-0AS01-0AA0	V2.6
OrderNumber:3RW4900-0NC0	
OrderNumber:6ES7 518-4AP00-0AB0	V41.0
OrderNumber:6ES7 155-5AA00-0AA0	
OrderNumber:6ES7 523-1BL00-0AA0	V2.0
OrderNumber:6ES7 534-7QE00-0AB0	V2.1
OrderNumber:6ES7 157-1AA00-0AB0	V4.0
OrderNumber:6ES7 144-5KD00-0BA0	V1.0
OrderNumber:6ES7 145-5ND00-0BA0	V1.0
OrderNumber:6ES7 143-5AF00-0BA0	V1.0
OrderNumber:6ES7 147-5JD00-0BA0	V1.0
OrderNumber:6ES7 314-6EH04-0AB0	V1.0
OrderNumber:6ES7 153-4BA00-0XB0	V1.0
OrderNumber:6GK7 343-2AH01-0XA0	V1.0
OrderNumber:6ES7 155-6BU00-0CN0	V3.3
OrderNumber:6ES7 131-6BF00-0BA0	V4.0
OrderNumber:6ES7 516-3AN01-0AB0	V3.1
OrderNumber:6EP1332-4BA00	V3.1
OrderNumber:6EP3436-8MB00-2CY0	V1.1
OrderNumber:6EP4293-8HB00-0XY0	V2.1
OrderNumber:6EP4297-8HB00-0XY0	
OrderNumber:6ES7 212-1BD30-0XB0	V1.1
OrderNumber:6EP4137-3AB00-2AY0	V1.1
OrderNumber:6EP4134-0GB00-0AY0	V1.1
OrderNumber:6ES7 516-3AN00-0AB0	V2.2
OrderNumber:6ES7 400-1TA01-0AA0	V2.2
OrderNumber:6ES7 405-0DA02-0AA0	V1.0
OrderNumber:6ES7 414-3EM05-0AB0	V1.8
OrderNumber:6ES7 412-1XJ05-0AB0	
OrderNumber:6GK7 443-1EX11-0XE0	
OrderNumber:6ES7 312-1AE13-0AB0	V5.3
OrderNumber:6GK7 343-1CX10-0XE0	V5.3
OrderNumber:6GK7 443-5DX04-0XE0	V2.7
OrderNumber:6GK7 342-5DA02-0XE0	V2.6
OrderNumber:6ES7 511-1AK00-0AB0	V2.4
OrderNumber:6GK7 542-1AX00-0XE0	V6.6
OrderNumber:6ES7 211-1BD30-0XB0	V5.0
OrderNumber:6GK7 242-7KX30-0XE0	V1.8

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 414-3EM05-0AB0	V1.0
OrderNumber:6ES7 151-1AA05-0AB0	V2.2
OrderNumber:6ES7 155-5AA00-0AC0	V1.0
OrderNumber:6ES7 155-6BU00-0CN0	V5.2
OrderNumber:6ES7 193-6PA00-0AA0	
OrderNumber:6ES7 505-0KA00-0AB0	V3.0
OrderNumber:6ES7 521-1BH50-0AA0	V3.0
OrderNumber:6ES7 511-1TK01-0AB0	V1.0
OrderNumber:6ES7 515-2TM01-0AB0	V1.0
OrderNumber:6ES7 521-1BH00-0AB0	V2.0
OrderNumber:6ES7 521-1BL10-0AA0	V2.1
OrderNumber:6ES7 521-7EH00-0AB0	V2.1
OrderNumber:6ES7 521-1FH00-0AA0	V2.1
OrderNumber:6ES7 522-5HH00-0AB0	V1.0
OrderNumber:6AG1 511-1AK00-7AB0	V1.0
OrderNumber:6AG1 505-0KA00-7AB0	V2.0
OrderNumber:6AG1 531-7KF00-7AB0	V1.0
OrderNumber:6AG1 531-7NF10-7AB0	V1.8
OrderNumber:6AG1 532-5HD00-7AB0	V1.0
OrderNumber:6AG1 532-5HF00-7AB0	V2.0
OrderNumber:6AG1 521-1BH00-7AB0	V2.0
OrderNumber:6AG1 521-1FH00-7AA0	V2.0
OrderNumber:6AG1 522-1BF00-7AB0	V2.0
OrderNumber:6AG1 522-5FF00-7AB0	V2.0
OrderNumber:6AG1 550-1AA00-7AB0	V2.0
OrderNumber:6ES7 522-1BH01-0AB0	V2.0
OrderNumber:6AG1 513-1AL00-2AB0	V2.0
OrderNumber:6AG1 541-1AB00-7AB0	V1.1
OrderNumber:6AG1 542-5DX00-7XE0	V1.0
OrderNumber:6ES7 505-0RA00-0AB0	V1.8
OrderNumber:6ES7 518-4AP00-3AB0	V1.0
OrderNumber:6GK7 542-1AX00-0XE0	V1.0
OrderNumber:6ES7 551-1AB00-0AB0	V1.0
OrderNumber:6ES7 531-7PF00-0AB0	V2.1
OrderNumber:6AG1 516-3AN00-7AB0	V2.0
OrderNumber:6ES7 517-3TP00-0AB0	V1.1
OrderNumber:6ES7 540-1AD00-0AA0	V1.1
OrderNumber:6ES7 532-5HD00-0AB0	V1.8
OrderNumber:6ES7 507-0RA00-0AB0	V2.1
OrderNumber:6ES7 515-2AM00-0AB0	V1.0
OrderNumber:7MH4 980-1AA01	V2.1
OrderNumber:7MH4 980-2AA01	V1.0
OrderNumber:6ES7 553-1AA00-0AB0	V1.8

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 552-1AA00-0AB0	
OrderNumber:6ES7 550-1AA00-0AB0	
OrderNumber:6AG1 516-3AN00-2AB0	V1.0
OrderNumber:6AG1 540-1AD00-7AA0	V1.0
OrderNumber:6AG1 540-1AB00-7AA0	V1.1
OrderNumber:6AG1 541-1AD00-7AB0	V1.8
OrderNumber:6ES7 511-1CK00-0AB0	V1.0
OrderNumber:6ES7 531-7NF10-0AB0	V1.0
OrderNumber:6ES7 532-5HF00-0AB0	V1.0
OrderNumber:6ES7 512-1CK00-0AB0	V2.1
OrderNumber:6ES7 521-1BH10-0AA0	V2.1
OrderNumber:6ES7 522-1BF00-0AB0	V2.1
OrderNumber:6ES7 212-1BE40-0XB0	V2.1
OrderNumber:6ES7 234-4HE32-0XB0	V1.0
OrderNumber:6ES7 223-1BH30-0XB0	V2.1
OrderNumber:6GK7 243-1BX30-0XE0	V4.2
OrderNumber:3RK7243-2AA30-0XB0	V2.0
OrderNumber:6ES7 217-1AG40-0XB0	V1.0
OrderNumber:6GT2 002-0LA00	V3.0
OrderNumber:6ES7 518-4FP00-0AB0	V1.1
OrderNumber:6ES7 517-3UP00-0AB0	V4.2
OrderNumber:6ES7 511-1FK00-0AB0	V1.0
OrderNumber:6ES7 531-7QD00-0AB0	V2.1
OrderNumber:6ES7 517-3AP00-0AB0	V2.1
OrderNumber:6ES7 522-5EH00-0AB0	V1.8
OrderNumber:6ES7 522-1BH10-0AA0	V1.0
OrderNumber:3SU1 400-1LK10-*AA1	V2.1
OrderNumber:3SU1 400-1MA10-1BA1	V1.0
OrderNumber:6EP3436-8MB00-2CY0	V1.0
OrderNumber:6EP4436-8XB00-0CY0	V1.0
OrderNumber:6EP4131-0GB00-0AY0	V1.2
GSD:SIEM804C.GSD/M/0	V1.2
OrderNumber:6ES7 143-2BH*0-0AB0	V1.0
OrderNumber:6ES7 143-2BH00-0AB0	
OrderNumber:6ES7 407-0DA02-0AA0	
OrderNumber:6ES7 460-0AA01-0AB0	
OrderNumber:6ES7 412-2XK07-0AB0	
OrderNumber:6ES7 450-1AP00-0AE0	
OrderNumber:6ES7 211-1AE31-0XB0	
OrderNumber:6ES7 241-1CH32-0XB0	V7.0
OrderNumber:6ES7 211-1HD30-0XB0	
OrderNumber:6ES7 241-1AH30-0XB0	V3.0
OrderNumber:6ES7 212-1AE31-0XB0	V2.2

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 221-1BF32-0XB0	V2.2
OrderNumber:6ES7 232-4HD30-0XB0	V1.0
OrderNumber:6ES7 510-1DJ01-0AB0	V3.0
OrderNumber:6ES7 512-1DK01-0AB0	V2.0
OrderNumber:6ES7 157-1AB00-0AB0	V1.0
OrderNumber:6ES7 512-1CK00-0AB0	V1.8
OrderNumber:6AG1 155-6AU00-7BN0	V1.8
OrderNumber:6AG1 193-6PA00-7AA0	V1.0
OrderNumber:6AG1 155-6AU00-7BN0	V1.8
OrderNumber:6ES7 313-6CF03-0AB0	V1.0
OrderNumber:6ES7 318-3EL00-0AB0	V1.0
OrderNumber:6ES7 151-3BA23-0AB0	V3.1
OrderNumber:3RK1 903-0BA00	V2.6
OrderNumber:3RK1 301-0BB13-1AA4	V2.7
OrderNumber:6ES7 518-4AP00-0AB0	V7.0
OrderNumber:6ES7 154-8AB00-0AB0	
OrderNumber:6ES7 144-4PF00-0AB0	
OrderNumber:6ES7 151-1BA02-0AB0	V1.6
OrderNumber:6ES7 131-4BB01-0AB0	V2.5
OrderNumber:6ES7 132-4BB01-0AA0	
OrderNumber:6ES7 131-4BB01-0AA0	
OrderNumber:6ES7 132-4BD00-0AB0	
OrderNumber:6ES7 407-0KA02-0AA0	
OrderNumber:6ES7 401-1DA01-0AA0	
OrderNumber:6ES7 416-3ES06-0AB0	
OrderNumber:6ES7 318-3EL01-0AB0	
OrderNumber:6ES7 138-4DA04-0AB0	
OrderNumber:6ES7 518-4FP00-3AB0	V6.0
OrderNumber:6ES7 317-2AJ10-0AB0	V3.2
OrderNumber:6ES7 132-4HB50-0AB0	V1.0
OrderNumber:6ES7 132-4BB31-0AA0	V2.0
OrderNumber:6ES7 416-3ER05-0AB0	V2.6
OrderNumber:6ES7 151-1AA04-0AB0	
GSD:SIEM818A.GSD/M/70000	
GSD:SIEM818A.GSD/M/13	V5.3
OrderNumber:6ES7 155-6AU00-0CN0	
OrderNumber:6ES7 412-2XJ05-0AB0	
OrderNumber:6ES7 431-7QH00-0AB0	
OrderNumber:6ES7 432-1HF00-0AB0	V3.3
OrderNumber:6ES7 460-1BA01-0AB0	V5.3
OrderNumber:6ES7 416-2XN05-0AB0	
GSD:SIEM818A.GSD/M/18	
OrderNumber:7MH4 138-6AA00-0BA0	

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 155-6AU00-0BN0	V5.3
OrderNumber:6ES7 513-1AL00-0AB0	
OrderNumber:6ES7 521-1BL00-0AB0	
OrderNumber:6ES7 522-1BL00-0AB0	V3.3
OrderNumber:6ES7 522-1BH00-0AB0	V1.8
OrderNumber:6ES7 540-1AB00-0AA0	V2.1
OrderNumber:6ES7 541-1AB00-0AB0	V2.0
OrderNumber:6GK7 542-5DX00-0XE0	V2.0
OrderNumber:6ES7 317-2EK14-0AB0	V1.0
OrderNumber:6ES7 516-3AN00-0AB0	V1.0
OrderNumber:6GK7 542-5FX00-0XE0	V2.0
OrderNumber:6GK7 543-1AX00-0XE0	V3.2
OrderNumber:6GK7 543-1AX00-0XE0	V1.1
OrderNumber:6ES7 516-3AN00-0AB0	V1.0
OrderNumber:6GK7 242-7KX30-0XE0	V1.0
OrderNumber:6ES7 211-1BE31-0XB0	V1.1
OrderNumber:6GK7 243-1JX30-0XE0	V1.5
OrderNumber:6GK7 243-5DX30-0XE0	V1.3
OrderNumber:6ES7 132-4BB01-0AB0	V3.0
OrderNumber:6ES7 152-1AA00-0AB0	V1.0
OrderNumber:6ES7 132-7GD00-0AB0	V1.0
OrderNumber:6ES7 334-0KE00-0AB0	
OrderNumber:6ES7 151-3BA23-0AB0	V2.0
OrderNumber:OPC Server	
OrderNumber:Application	
OrderNumber:6ES7 214-1BG40-0XB0	V6.0
OrderNumber:6ES7 221-1BH32-0XB0	SW V14 ...
OrderNumber:6ES7 222-1XF30-0XB0	SW V8.1 SP2 ...
OrderNumber:6ES7 223-1QH30-0XB0	V4.2
OrderNumber:6ES7 231-4HF30-0XB0	V2.0
OrderNumber:6ES7 234-4HE30-0XB0	V1.0
OrderNumber:7MH4960-2AA01	V1.0
OrderNumber:6ES7 214-1AG40-0XB0	V1.0
OrderNumber:6ES7 228-1RC52-0AA0	V1.0
OrderNumber:6ES7 278-4BD32-0XB0	
OrderNumber:6ES7 214-1HG31-0XB0	V4.2
OrderNumber:7MH4960-6AA01	V2.2
OrderNumber:7MH4960-4AA01	V2.0
OrderNumber:6ES7 223-1PL30-0XB0	V3.0
OrderNumber:6ES7 215-1BG40-0XB0	
OrderNumber:6ES7 231-4HD30-0XB0	
OrderNumber:6ES7 231-5PD32-0XB0	V1.0
OrderNumber:6ES7 231-5QF30-0XB0	V4.2

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 232-4HB32-0XB0 OrderNumber:3RK1 301-0CB10-1AB4	V1.0 V2.0
OrderNumber:3RK1 903-0CK00	V1.0
OrderNumber:6ES7 215-1HF40-0XB0	V2.0
OrderNumber:6ES7 212-1AF40-0XB0	V10.0
OrderNumber:6ES7 214-1HF40-0XB0	
OrderNumber:6ES7 214-1AF40-0XB0	V4.2
OrderNumber:6ES7 515-2FM00-0AB0	V4.2
OrderNumber:6ES7 511-1UK01-0AB0	V4.2
OrderNumber:6ES7 215-1AG31-0XB0	V4.2
OrderNumber:6ES7 221-1BF30-0XB0	V1.8
OrderNumber:6ES7 222-1XF32-0XB0	V2.1
OrderNumber:6ES7 222-1HF30-0XB0	V3.0
OrderNumber:6ES7 223-1BL30-0XB0	V1.0
OrderNumber:6ES7 215-1HG40-0XB0	V2.0
OrderNumber:6ES7 241-1CH31-0XB0	V1.0
OrderNumber:6ES7 212-1HF40-0XB0	V1.0
OrderNumber:6AG1 214-1AG40-4XB0	V4.2
OrderNumber:6AG1 221-1BF32-2XB0	V1.0
OrderNumber:6AG1 221-1BF32-4XB0	V4.2
OrderNumber:6AG1 222-1BH32-2XB0	V4.2
OrderNumber:6AG1 222-1BF32-4XB0	V2.0
OrderNumber:6AG1 223-1BL32-2XB0	V2.0
OrderNumber:6AG1 214-1HG40-2XB0	V2.0
OrderNumber:6AG1 223-1PH32-2XB0	V2.0
OrderNumber:6AG1 223-1PL32-4XB0	V2.0
OrderNumber:6AG1 223-1QH32-4XB0	V4.2
OrderNumber:6AG1 222-1XF32-2XB0	V2.0
OrderNumber:6AG1 222-1HF32-4XB0	V2.0
OrderNumber:6AG1 222-1HH32-2XB0	V2.0
OrderNumber:6AG1 215-1BG40-5XB0	V2.0
OrderNumber:6AG1 231-5QF32-4XB0	V2.0
OrderNumber:6AG1 231-4HD32-4XB0	V2.0
OrderNumber:6AG1 222-1BF32-2XB0	V4.2
OrderNumber:6AG1 234-4HE32-2XB0	V2.0
OrderNumber:6AG1 278-4BD32-4XB0	V2.0
OrderNumber:6AG1 215-1AG40-4XB0	V2.0
OrderNumber:6AG1 221-1BH32-2XB0	V2.0
OrderNumber:6AG1 241-1CH32-2XB0	V2.0
OrderNumber:6AG1 215-1HG40-2XB0	V4.2
OrderNumber:6ES7 223-1PL32-0XB0	V2.0
OrderNumber:6ES7 228-1RC51-0AA0	V2.1
OrderNumber:6ES7 238-5XA32-0XB0	V4.2

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 215-1AF40-0XB0	V2.0
OrderNumber:6ES7 223-1BL32-0XB0	V2.2
OrderNumber:6AG1 212-1BE40-2XB0	V2.0
OrderNumber:6AG1 243-5DX30-2XE0	V4.2
OrderNumber:6AG1 212-1HE40-2XB0	V2.0
OrderNumber:6AG1 223-1QH32-2XB0	V4.2
OrderNumber:6EP3436-8SB00-2AY0	V1.3
OrderNumber:6EP3437-8MB00-2CY0	V4.2
OrderNumber:6EP4137-3AB00-2AY0	V2.0
OrderNumber:6EP4136-3AB00-2AY0	V1.1
OrderNumber:6EP4133-0GB00-0AY0	V1.1
OrderNumber:6ES7 138-4DC01-0AB0	V2.1
System:Port.Scalance/Comboport_MAU	V2.1
OrderNumber:6GK1 160-4AT01	V1.0
OrderNumber:6GK1 160-4AA00	
OrderNumber:6GK1 161-6AA02	
OrderNumber:6GK1 551-2AA00	V2.7
OrderNumber:6GK1 571-1AA00	V2.6
OrderNumber:6GK1 562-3AA00	V2.7
OrderNumber:6GK1 561-3AA02	SW V6.1 ...
OrderNumber:6GK7 343-1GX31-0XE0	SW V7.1 SP1 ...
OrderNumber:6GK7 443-5DX05-0XE0	SW V7.1 SP2 ...
OrderNumber:6ES7 214-1BE30-0XB0	SW V12 ...
OrderNumber:6ES7 241-1AH32-0XB0	V3.0
OrderNumber:6ES7 211-1AD30-0XB0	V7.0
OrderNumber:6ES7 231-4HA30-0XB0	V2.2
OrderNumber:6ES7 214-1HG40-0XB0	V2.2
OrderNumber:6GK7 242-7KX30-0XE0	V2.2
OrderNumber:6ES7 212-1AD30-0XB0	V2.0
OrderNumber:6ES7 138-4GA50-0AB0	V4.0
OrderNumber:6ES7 132-4BF00-0AB0	V1.4
OrderNumber:6ES7 154-4AB10-0AB0	V2.0
OrderNumber:6ES7 154-8FB01-0AB0	
OrderNumber:6ES7 155-6AU01-0BN0	
OrderNumber:6ES7 155-6AU00-0DN0	V7.1/Cu
OrderNumber:6ES7 155-5AA00-0AB0	V3.2
OrderNumber:6ES7 155-5AA01-0AB0	V4.1
OrderNumber:6AG1 155-5AA00-2AC0	V4.0
OrderNumber:6ES7 155-5BA00-0AB0	V3.0
OrderNumber:6AG1 155-5BA00-2AB0	V4.1
OrderNumber:6ES7 518-4AP00-0AB0	V3.0
OrderNumber:6ES7 132-6BF00-0AA0	V3.0
OrderNumber:6ES7 132-6HD00-0BB0	V3.0

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 132-6BD20-0CA0	V2.5
OrderNumber:6ES7 134-6PA01-0BD0	V1.0
OrderNumber:6ES7 132-6HD00-0BB1	V1.1
OrderNumber:6ES7 531-7NF00-0AB0	V2.0
OrderNumber:6ES7 223-0BD30-0XB0	V3.0
OrderNumber:6ES7 212-1AE40-0XB0	V1.1
OrderNumber:6ES7 221-3AD30-0XB0	V1.1
OrderNumber:6ES7 214-1AE30-0XB0	V1.0
OrderNumber:6ES7 222-1BD30-0XB0	V4.2
OrderNumber:6ES7 222-1AD30-0XB0	V1.0
OrderNumber:6ES7 232-4HA30-0XB0	V2.2
OrderNumber:6ES7 416-3FR05-0AB0	V'.0
OrderNumber:6ES7 964-2AA04-0AB0	V1.0
OrderNumber:6ES7 414-3FM07-0AB0	V1.0
OrderNumber:6ES7 416-2FP07-0AB0	V5.3
OrderNumber:6ES7 151-8FB01-0AB0	
OrderNumber:6ES7 151-7FA20-0AB0	V7.0
OrderNumber:6ES7 132-7RD22-0AB0	V7.0
OrderNumber:6ES7 132-7GD21-0AB0	V3.2
OrderNumber:6ES7 315-2EH13-0AB0	V2.6
OrderNumber:6ES7 322-8BH10-0AB0	
OrderNumber:7MH4 900-2AA01	
OrderNumber:7MH4 900-3AA01	V2.6
OrderNumber:7MH4 950-1AA01	
OrderNumber:7MH4 950-2AA01	
OrderNumber:7MH4920-0AA01	
OrderNumber:7MH4910-0AA01	
OrderNumber:6AG1 155-5AA00-7AB0	
OrderNumber:6AG1 505-0RA00-7AB0	
OrderNumber:6AG1 522-1BL00-7AB0	
OrderNumber:3SU1 401-1MC*0-1CA1	V3.0
OrderNumber:3SU1 401-1ME*0-1DA1	V1.0
OrderNumber:6BK1900-0AA00-0AA*	V1.0
OrderNumber:6BK1900-0BA00-0AA*	V1.0
OrderNumber:6BK1900-0CA00-0AA*	V1.0
OrderNumber:6BK1942-2AA00-0AA*	/HCS4200
OrderNumber:6ES7 412-2EK06-0AB0	/HCS4200
OrderNumber:6GK7 343-2AH11-0XA0	/HCS4200
OrderNumber:6ES7 313-6CG04-0AB0	
OrderNumber:6GK5 991-2AB00-8AA0	V6.0
OrderNumber:6GK5 992-2VA00-8AA0	V3.1
OrderNumber:6GK5 992-2AS00-8AA0	V3.3
OrderNumber:6GK5 400-8AS00-8AP2	

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6GK5 905-0PA00	
OrderNumber:6GK5 991-1AD00-8AA0	/legacy
OrderNumber:6GK5 992-1AQ00-8AA0	
OrderNumber:6GK5 907-4PA00	
OrderNumber:6ES7 143-2BH50-0AB0	
OrderNumber:6ES7 405-0KA02-0AA0	
OrderNumber:6ES7 513-1FL01-0AB0	
OrderNumber:6ES7 134-7TD00-0AB0	
OrderNumber:6ES7 516-3FN00-0AB0	
OrderNumber:6ES7 151-8AB01-0AB0	V2.1
OrderNumber:3RK1 301-0BB10-1AA4	
OrderNumber:6ES7 151-1CA00-3BL0	V1.8
OrderNumber:3RK1 301-0CB13-0AA4	V3.2
OrderNumber:6ES7 315-2EH14-0AB0	
OrderNumber:6ES7 321-1BH02-0AA0	
OrderNumber:6ES7 421-7BH01-0AB0	
OrderNumber:6ES7 138-4HA00-0AB0	V3.2
OrderNumber:6SL3 244-0SA00-1AA1	
OrderNumber:6SL3 244-0SA01-1AA1	
OrderNumber:6GT2 002-0ED00	V1.0
OrderNumber:6ES7 135-7TD00-0AB0 ES9	3.0
OrderNumber:6ES7 340-1CH02-0AE0	3.0
OrderNumber:6ES7 352-1AH02-0AE0	V5.0/GENERIC
OrderNumber:6ES7 400-1TA11-0AA0	
OrderNumber:6ES7 416-3XR05-0AB0	V1.0
OrderNumber:3RK1 301-0AB10-1AA3	
OrderNumber:6ES7 672-7AC01-0YA0	
OrderNumber:6ES7 307-1BA01-0AA0	V5.3
OrderNumber:6ES7 351-1AH01-0AE0	
OrderNumber:6ES7 334-0CE01-0AA0	V2.1
OrderNumber:6ES7 312-5BF04-0AB0	
OrderNumber:6ES7 313-5BG04-0AB0	
OrderNumber:6ES7 338-7XF00-0AB0 IDENT	
OrderNumber:6ES7 314-6CH04-0AB0	
OrderNumber:6ES7 315-2AH14-0AB0	V3.3
OrderNumber:7ME4 120-2DH21-0EA0	V3.3
OrderNumber:6ES7 322-1CF00-0AA0	
OrderNumber:6ES7 335-7HG02-0AB0	V3.3
OrderNumber:6ES7 338-4BC01-0AB0	V3.3
OrderNumber:6ES7 355-2CH00-0AE0	
OrderNumber:7ME4 120-2DH20-0EA0	
OrderNumber:6ES7 321-1CH20-0AA0	
OrderNumber:6ES7 321-1CH00-0AA0	V3.0

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 322-8BF00-0AB0	
OrderNumber:6ES7 322-1BL00-0AA0	
OrderNumber:6ES7 322-1FF01-0AA0	
OrderNumber:6ES7 305-1BA80-0AA0	
OrderNumber:6ES7 341-1BH01-0AE0	
OrderNumber:6ES7 331-7KB02-0AB0	
OrderNumber:6ES7 331-7KF02-0AB0	
OrderNumber:6ES7 331-1KF01-0AB0	
OrderNumber:6ES7 332-5HB01-0AB0	V1.0
OrderNumber:6ES7 332-5HD01-0AB0	
OrderNumber:6ES7 332-5HF00-0AB0	
OrderNumber:6ES7 321-7RD00-0AB0	
OrderNumber:6ES7 321-1FH00-0AA0	
OrderNumber:6ES7 322-1HH01-0AA0	
OrderNumber:6ES7 322-5HF00-0AB0	
OrderNumber:6ES7 322-1HF10-0AA0	
OrderNumber:6ES7 327-1BH00-0AB0	
OrderNumber:6ES7 351-1AH02-0AE0	
OrderNumber:6ES7 321-7BH01-0AB0	
OrderNumber:6ES7 322-1BH01-0AA0	
OrderNumber:6ES7 322-5GH00-0AB0	
OrderNumber:6ES7 321-1FF01-0AA0	
OrderNumber:6GK7 443-1EX41-0XE0	
OrderNumber:6ES7 414-2XK05-0AB0	
OrderNumber:6ES7 431-1KF00-0AB0	
OrderNumber:6ES7 414-3XM05-0AB0	
OrderNumber:6ES7 421-1EL00-0AA0	V1.0
OrderNumber:6ES7 414-3EM07-0AB0	V5.3
OrderNumber:6ES7 416-3XS07-0AB0	
OrderNumber:6ES7 414-3FM06-0AB0	V5.3
OrderNumber:6ES7 421-1FH20-0AA0	
OrderNumber:6ES7 416-3FS07-0AB0	V7.0
OrderNumber:6ES7 431-0HH00-0AB0	V7.0
OrderNumber:6ES7 416-2FN05-0AB0	V6.0
OrderNumber:6ES7 455-0VS00-0AE0	
OrderNumber:6ES7 417-4XT07-0AB0	V7.0
OrderNumber:6ES7 421-1BL01-0AA0	
OrderNumber:6ES7 422-1BL00-0AA0	V5.3
OrderNumber:6ES7 452-1AH01-0AE0	
OrderNumber:6ES7 422-1BH11-0AA0	V7.0
OrderNumber:6ES7 421-7DH00-0AB0	
OrderNumber:6ES7 422-1FH00-0AA0	
OrderNumber:6ES7 422-1HH00-0AA0	

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 452-1AH00-0AE0	
OrderNumber:6ES7 455-1VS00-0AE0	
OrderNumber:6ES7 450-1AP01-0AE0	
OrderNumber:6GK7 443-5FX02-0XE0	
OrderNumber:6ES7 517-3FP00-0AB0	
OrderNumber:6ES7 515-2UM01-0AB0	
OrderNumber:6ES7 531-7KF00-0AB0	
OrderNumber:6ES7 211-1BE40-0XB0	V4.0
OrderNumber:6AG4114-2yxxx-xxxx//Device	V2.1
OrderNumber:6AG4112-2yxxx-xxxx//Device	V2.1
OrderNumber:6AG4114-2zxxx-xxxx//Device	V2.1
OrderNumber:6AG4132-2yxxx-xxxx//DeviceOrderNumber:6AV7884-0xyxx-xxxx//Device.12inch.Touch	V4.2
OrderNumber:6ES7647-7Byxx-xxxx//Device OrderNumber:6AV7884-5xyxx-xxxx//Device.19inch.Touch	
OrderNumber:6AV7884-2xyxx-xxxx//Device.15inch.Touch	
OrderNumber:6ES7647-7Byxx-xxxx//Device	
OrderNumber:6ES7647-7Bzxx-xxxx//Device OrderNumber:6AV7853-0xzxx-xxxx//Device.15inch.Touch OrderNumber:6AV7854-0xzxx-xxxx//Device.15inch.Key OrderNumber:6AV7854-0xxxx-xxxx//Device.15inch.Key OrderNumber:6ES7647-7Ayxx-xxxx//Device OrderNumber:6AV7856-0xyxx-xxxx//Device.19inch.Touch OrderNumber:6ES7647-7Ayxx-xxxx//Device OrderNumber:6AV7873-xxxxx-xBxx//Device.15inch.Key OrderNumber:6ES7643-8yxxx-xxxx//Device OrderNumber:6ES7647-6Byxx-xxxx//Device OrderNumber:6AV7870-xxxxx-xAxx//Device.12inch.Touch OrderNumber:6AV7872-xxxxx-xAxx//Device.15inch.Touch OrderNumber:6AV7874-xxxxx-xBxx//Device.17inch.Touch	
OrderNumber:6AV7875-xxxxx	
-xAxx//Device.19inch.Touch	
OrderNumber:6AG4112-0yxxx-xxxx//Device	
ET200eco - HeadModules	
ET200eco PN - HeadModules	
SIMATIC RF600 - HeadModules	
OrderNumber:3RK1 304-5KS40-2AA3	
OrderNumber:3RK1 304-5LS40-2AA3	
OrderNumber:3RK1 304-0HS00-8AA0	
OrderNumber:3RK1 304-0HS00-6AA0	
OrderNumber:6ES7 516-2PN00-0AB0	
OrderNumber:6ES7 151-3BB23-0AB0	
OrderNumber:6ES7 154-6AB00-0AB0	
OrderNumber:6ES7 141-5AH00-0BA0	
OrderNumber:6ES7 142-5AF00-0BA0	
OrderNumber:6ES7 143-5AH00-0BA0	

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6GK7 243-8RX30-0XE0	V3.0
OrderNumber:6ES7 212-1HE40-0XB0	V3.0
OrderNumber:6ES7 518-4FP00-0AB0	V1.0
OrderNumber:6AG1 215-1HG40-4XB0	V1.0
OrderNumber:6ES7 138-4FD00-0AA0	V2.1
OrderNumber:6ES7 134-4NB51-0AB0	V7.0
OrderNumber:6ES7 135-4GB52-0AB0	V1.0
OrderNumber:6ES7 138-7AA00-0AA0	V1.0
OrderNumber:6ES7 138-7BB00-0AB0	V1.0
OrderNumber:6ES7 145-4GF00-0AB0	V1.0
OrderNumber:6ES7 143-4BF50-0AA0	V3.0
OrderNumber:6GK5 980-3CB00-0AA1	V4.2
OrderNumber:6GK5 980-3CB00-0AA7	V2.1
OrderNumber:6GK5 991-4AB00-8AA0	V4.2
OrderNumber:6GK5 992-4AL00-8AA0	
OrderNumber:6GK5 992-4SA00-8AA0	
OrderNumber:6GK5 992-4RA00-8AA0	
OrderNumber:6GK5 992-4GA00-8AA0	
OrderNumber:6GK5 991-1AE00-8AA0	
OrderNumber:6GK5 991-1AF00-8AA0	
OrderNumber:6GK5 992-1AL00-8AA0	
OrderNumber:6GK5 992-1AN00-8AA0	
OrderNumber:6ES7 174-0AA10-0AA0	
OrderNumber:6GK5 204-0BA00-2BF2	
OrderNumber:6NH9 741-1AA00	
OrderNumber:6ES7 133-1BL01-0XB0	
OrderNumber:6ES7 132-1BL00-0XB0	
OrderNumber:6ES7 132-1BH00-0XB0	
OrderNumber:6ES7 131-1BL01-0XB0	
OrderNumber:6ES7 131-1BH01-0XB0	
OrderNumber:3RK1 207-2BQ44-0AA3 ET200eco - HeadModules	
ET200eco PN - HeadModules	
SIMATIC RF600 - HeadModules	
OrderNumber:3RK1 304-5KS40-2AA3	
OrderNumber:3RK1 304-5LS40-2AA3	
OrderNumber:3RK1 304-0HS00-8AA0	
OrderNumber:3RK1 304-0HS00-6AA0	
OrderNumber:6ES7 516-2PN00-0AB0	
OrderNumber:6ES7 151-3BB23-0AB0	
OrderNumber:6ES7 154-6AB00-0AB0	
OrderNumber:6ES7 141-5AH00-0BA0	
OrderNumber:6ES7 142-5AF00-0BA0	

Whitelist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6ES7 143-5AH00-0BA0	
OrderNumber:6GK7 243-8RX30-0XE0	
OrderNumber:6ES7 212-1HE40-0XB0	
OrderNumber:6ES7 518-4FP00-0AB0	
OrderNumber:6AG1 215-1HG40-4XB0	
OrderNumber:6ES7 138-4FD00-0AA0	
OrderNumber:6ES7 134-4NB51-0AB0	
OrderNumber:6ES7 135-4GB52-0AB0	
OrderNumber:6ES7 138-7AA00-0AA0	
OrderNumber:6ES7 138-7BB00-0AB0	
OrderNumber:6ES7 145-4GF00-0AB0	
OrderNumber:6ES7 143-4BF50-0AA0	
OrderNumber:6GK5 980-3CB00-0AA1	
OrderNumber:6GK5 980-3CB00-0AA7	
OrderNumber:6GK5 991-4AB00-8AA0	
OrderNumber:6GK5 992-4AL00-8AA0	
OrderNumber:6GK5 992-4SA00-8AA0	
OrderNumber:6GK5 992-4RA00-8AA0	
OrderNumber:6GK5 992-4GA00-8AA0	
OrderNumber:6GK5 991-1AE00-8AA0	
OrderNumber:6GK5 991-1AF00-8AA0	
OrderNumber:6GK5 992-1AL00-8AA0	
OrderNumber:6GK5 992-1AN00-8AA0	
OrderNumber:6ES7 174-0AA10-0AA0	
OrderNumber:6GK5 204-0BA00-2BF2	
OrderNumber:6NH9 741-1AA00	
OrderNumber:6ES7 133-1BL01-0XB0	
OrderNumber:6ES7 132-1BL00-0XB0	
OrderNumber:6ES7 132-1BH00-0XB0	
OrderNumber:6ES7 131-1BL01-0XB0	
OrderNumber:6ES7 131-1BH01-0XB0	
OrderNumber:3RK1 207-2BQ44-0AA3	
	V1.1
	V5.3
	V1.1

Below table shows list of device/deviceitems that does not support import using library references.

Blacklist	
TypeIdentifier	**FirmwareVersion**
OrderNumber:6AG1 193-6AR00-7AA0	V0.0
OrderNumber:6ES7 193-6AR00-0AA0	V0.0
OrderNumber:6GK5 991-2VA00-8AA2	V0.0
OrderNumber:6ES7 193-6AM00-0AA0	V0.0
OrderNumber:6ES7 193-6AF00-0AA0	V0.0
OrderNumber:6ES7 193-6AP00-0AA0	V0.0
OrderNumber:6ES7 193-6AP20-0AA0	V0.0
OrderNumber:6ES7 193-6AP40-0AA0	V0.0
OrderNumber:6ES7 193-6AG00-0AA0	V0.0
OrderNumber:6ES7 193-6AG20-0AA0	V0.0
OrderNumber:6ES7 193-6AG40-0AA0	V0.0
OrderNumber:6ES7 193-6AP40-0AA0	V0.0
OrderNumber:6DL1 193-6AG00-0AA0	V0.0
OrderNumber:6DL1 193-6AF00-0AA0	V0.0
OrderNumber:6DL1 193-6AR00-0AA0	V0.0
OrderNumber:6DL1 193-6AG20-0AA0	V0.0
OrderNumber:6DL1 193-6AG40-0AA0	V0.0
OrderNumber:6ES7 655-5PX11-0XX0	V1.2
SIMATIC COMMUNICATION MODULES - HeadModules	
SIMATIC optical identification systems - HeadModules	
OrderNumber:6AV6 646-1AC22-0AX0	
OrderNumber:6GK5 826-2AB00-2AB2	
OrderNumber:6GK5 991-1AD00-8FA0	V2.0
SCALANCE XC208 - HeadModules	V5.0
SCALANCE XB208- HeadModules	
SENTRON - HeadModules	
OrderNumber:6GK5 992-4AS00-8AA0	
Additional Ethernet devices	
Industrial PCs	
PCU 50.5	

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.24 Export of a device object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The PLC is offline.

Application

The `Device` object is a container object for a central or distributed configuration. It is the parent object for `DeviceItem` objects and the top level internal element of the instance hierarchy of the TIA Portal project in between an AML file's structure.

The CAx data export supports the following types of devices specified via the AML type identifier:

- Physical modules
- GSD/GSDML based devices
- Systems

Devices can be grouped in a `DeviceUserFolder` object.

Note

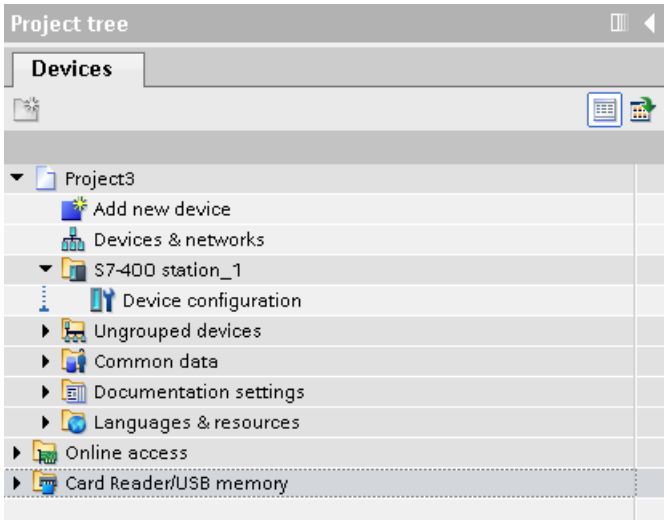
Export of a single device also exports all subnets in the project.

Attributes

The following table shows the related attributes of the device object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	
TypeIdentifier	Mandatory for export	Optional for import starting from AR APC V1.1
Comment	Optional	Default: ""

Example: Exported Configuration



AML structure of the export file

The following structure example depicts the export of the single device "S7-400 station_1" without racks and modules:

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="288b7850-688e-43b3-941e-d615ba900a02" Name="Project3">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="57611cfd-6da4-444e-ac78-5fbcea20a4e1" Name="S7-400 station_1">
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Device.S7400</Value>
      </Attribute>
      <Attribute Name="Comment" AttributeDataType="xs:string">
        <Value>S7400 station</Value>
      </Attribute>
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
  </InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

Device with no type identifier and generic type identifier

The CAx import shall be able to handle devices having no type Identifier information or having generic type identifiers ('System:Device.Generic'). It shall be possible that AMl file contains certain type of device with no type Identifier information or generic type identifier ('System:Device.Generic').

But CAx import shall handle these as well to create proper devices.

The following devices support generic device or no type identifier substitution:

- GSD and GSDML devices
- Devices based on MDD (Non-GSD/GSDML devices)

For Generic device or device having no type identifier and Generic Rack, type identifier replacement , it is mandatory that the head module (in case of decentral devices) or PLC (in case of central devices) has to be present in the rack described in the AML file, otherwise the device with no type identifiers or generic device and generic rack type identifier substitution shall fail.

The following XML structure shows a device configuration with non generic type identifier:

```
<InternalElement ID="04f5d5f08-316a-4a1d-9290-9bfd75b2b2ca" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 station</Value>
  </Attribute>
 <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device"/>
</InternalElement>
```

The following XML structure shows a device configuraton with generic type identifier:

```
<InternalElement ID="04f5d5f08-316a-4a1d-9290-9bfd75b2b2ca" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.Generic</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
 <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device"/>
</InternalElement>
```

The following XML structure shows a device configuration with type identifier available for device.

```
<InternalElement ID="a887601f-3ced-4f50-88ff-a9ec6eabb682" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
 <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device"/>
</InternalElement>
```

The following XML structure shows a device configuration with No Type Identifier available for device.

```
<InternalElement ID="a887601f-3ced-4f50-88ff-a9ec6eabb682" Name="S7-400 station_1">
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
 <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device"/>
</InternalElement>
```

See also

Structure of the CAx data for importing/exporting (Page 1435)

AML type identifiers (Page 1439)

6.5.25 Import of a device object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The PLC is offline.

Application

The `Device` object is a container object for a central or distributed configuration. It is the parent object for `DeviceItem` objects and the top level internal element of the instance hierarchy of the TIA Portal project in between an AML file's structure.

The CAx data import supports the following types of devices specified via the AML type identifier:

- Physical modules
- GSD/GSDML based devices
- Systems
- Generic devices

If a `DeviceUserFolder` object exists in the TIA Portal project, the devices will be grouped in the respective folder.

If you only know the identification (`TypeIdentifier`) of a head module or a PLC and not of the respective rack and device, you can import a generic rack.

Example: `TypeIdentifier = System:<Prefix>.Generic`

For generic device replacement, the following elements have to be present in the rack described in the AML file:

- Central devices: PLC
- Decentral devices: Head module

If devices are generic, the attribute `BuiltIn` defines the kind of rack or module:

- physical: `BuiltIn = True`
- generic: `BuiltIn = False`

Example: Importing a generic device

The following structure example depicts the import of the generic "S7-400 station" device without racks and modules.

```
...
  <InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="MyProject">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="3e6277d1-1b12-4c18-b00e-25e3eac3ac35" Name="S7400 station_1">
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Device.Generic</Value>
      </Attribute>
      <Attribute Name="Comment" AttributeDataType="xs:string">
        <Value>S7400 station_1</Value>
      </Attribute>
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
     ...
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
  </InternalElement>
 </InstanceHierarchy>
</CAEXFile>
```

Example: Importing a device user folder hierarchy

The following structure example depicts the import of a folder hierarchy.

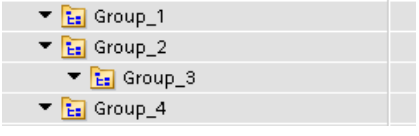
```
...
<InternalElement ID="4fe37f4f-2661-4492-95f0-3d8a8160c851" Name="Project1">
      <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
      <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
      <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
      <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
      <InternalElement ID="1ee1615f-9c67-432d-a7cc-b795babf67b6" Name="Group_1">
        <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
      </InternalElement>
      <InternalElement ID="ce14c85a-28de-41aa-ad08-2eb7d0fb755f" Name="Group_2">
        <InternalElement ID="852347e8-3c48-4eb9-8bd8-349d0c7caf34" Name="Group_3">
          <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
        </InternalElement>
        <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
      </InternalElement>
      <InternalElement ID="97cf7924-1756-4e32-8716-ac18990e4762" Name="Group_4">
        <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
      </InternalElement>
      <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
    </InternalElement>
...
```

Imported user folder hierarchy

The name of the folders for ungrouped and unassignd devics is language specific. It is recommended to perform an import with the same user interface language as the export. Otherwise ungrouped and unassigned devices will be imported into folders named according to the export language.

For example, if you have exported a project which contains Device System Group "Ungrouped devices" in English language and then import the AML file in Germany language. You will see that project should have a "Nicht gruppierte Gerate" (German language) exists in device system group and have "Ungrouped device" created as user group while CAx import.

The following hierarchy is imported into the project navigation:



See also

Structure of the CAx data for importing/exporting (Page 1435)

AML type identifiers (Page 1439)

6.5.26 Export/Import of device with set address

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

In TIA Portal, you can export the address objects of IO device items to an AML file. While importing to an empty TIA Portal project, the imported device items retains the address objects in its respective IO device items.

The Address attribute in the AML file contains RefSemantic mandatorily set to the specified value named OrderedListType.

Up to TIA Portal V16, It is expected that addresses should be in same order to support export and import of device item. For example, if a device item supports two addresses (Input and Output) in TIA Portal and same device item in AML file comes with addresses (Output and Input), addresses are not processed due to mismatch in order.

From TIA Portal V17, CAx import operation, for a particular address from AML file shall find the matching TIA address based on I/O type, and shall try to set the same. The order in the AML file shall not matter any more.

Attributes of a "Address" element

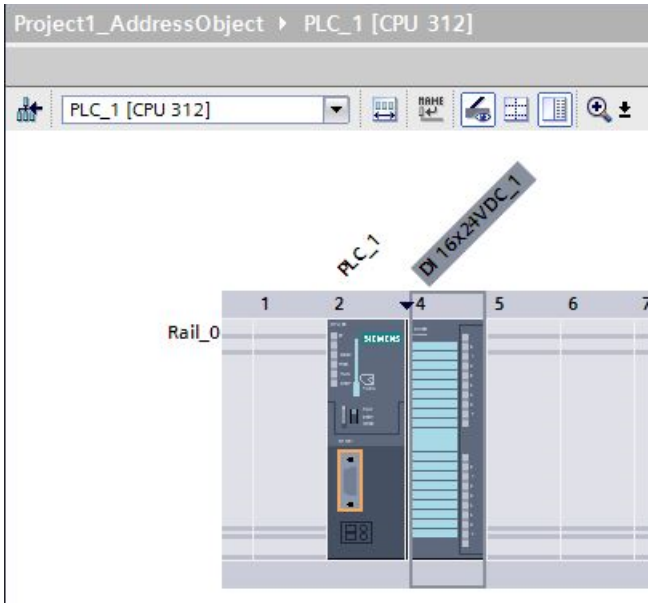
The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
IoType	Mandatory	Input or Output.
Length	Optional	Channel width. This attribute shall always be written to file during export
StartAddress	Optional	Start address of the IO device. This attribute shall always be written to file during export

Note

- The feature supports any type of device item with an address object of Input/Output type
- Input and Output addresses of length 0 shall be excluded from export and ignored at import
- Export of address attribute shall be skipped for the addresses having startaddress as -1. However during import, warning message shall be shown for the addresses having startaddress as -1

Example: IO device items with address objects



AML Structure

The following figures shows a partial element structure of the exported AML file. It contains the Address elments and its attributes.

```
<Attribute Name="Address">
<RefSemantic CorrespondingAttributePath="OrderedListType" />
<Attribute Name="1">
<Attribute Name="StartAddress" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>16</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
</Attribute>
```

Pruned XML

Pruning is the act of optimizing the content by removing certain things which are not necessarily to be provided in the XML. In this reduced xml, the auto created sub module information are not be provided, and its corresponding address object are provided under the immediate parent.

The following figure shows a partial element structure of the exported AML file before pruning.

```
<InternalElement ID="5511a117-42c6-44b7-be5d-0f33cd46e932" Name="AS-i Master_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>4</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>True</Value>
</Attribute>
<Attribute Name="Address">
<RefSemantic CorrespondingAttributePath="OrderedListType"/>
<Attribute Name="1">
<Attribute Name="StartAddress" AttributeDataType="xs:int">
<Value>20</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>256</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
```

In the pruned AML file, the sub module information like <InternalElement> element is removed and its corresponding address objects are retained.

```
<Attribute Name="Address">
<RefSemantic CorrespondingAttributePath="OrderedListType"/>
<Attribute Name="1">
<Attribute Name="StartAddress" AttributeDataType="xs:int">
<Value>20</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>256</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
</Attribute>
```

See also

Pruned AML (Page 1431)

6.5.27 Export/Import of device with channels

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

In TIA Portal, you can export the channel objects of IO device items to an AML file. While importing to an empty TIA Portal project, the imported device items retains the channel objects in its respective IO device items.

<ExternalInterface> element represents in node and subnet internal elements, and indicates that nodes and subnets are connected.

Attributes of a "ExternalInterface" element

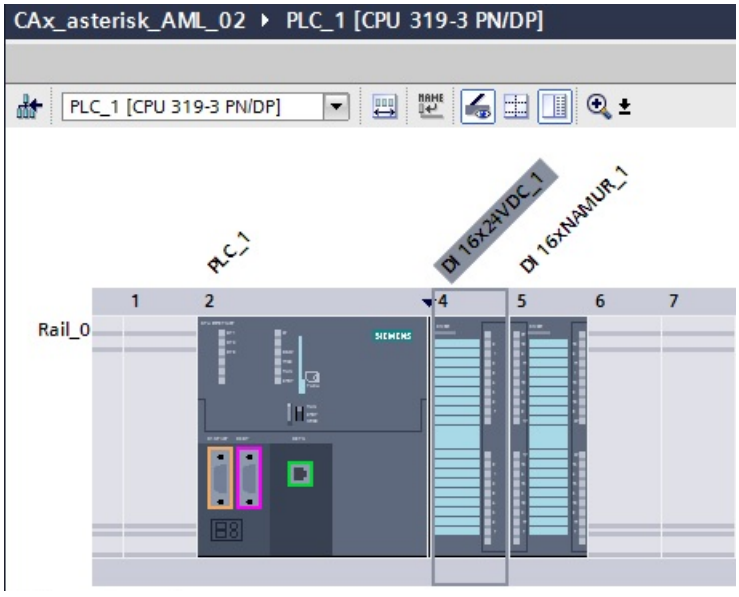
The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
IoType	Mandatory	Input or Output
Length	Optional	Channel width (1 for Digitial and 16 for Analog signals)
Number	Mandatory	Channel number starts from 0
Type	Mandatory	Analog or Digital

Channel numbering

Digital Input,Digitial Output, Analog Input, Analog Ouput, and Technology channels are numbered as DI_0, DO_0, AI_0, AO_0,TO_0 respectively. Channels on the device items itself are numbered first, and channels on sub-device items are numbered subsequently(depth first). Every additional device item has its own channel numbers starting from 0.

Example: Devices with channels



AML structure

The following figures shows a partial element structure of the exported AML file.

```
<ExternalInterface ID="31ca16d3-6322-43b6-95bc-e2d7d7bfc7b7" Name="Channel_DI_0"
                RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
          <Attribute Name="Type" AttributeDataType="xs:string">
            <Value>Digital</Value>
          </Attribute>
          <Attribute Name="IoType" AttributeDataType="xs:string">
            <Value>Input</Value>
          </Attribute>
          <Attribute Name="Number" AttributeDataType="xs:int">
            <Value>0</Value>
          </Attribute>
          <Attribute Name="Length" AttributeDataType="xs:int">
            <Value>1</Value>
          </Attribute>
        </ExternalInterface>
```

6.5.28 Export of device item objects

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The PLC is offline.

Application

The export of device item objets is only applicable for PLC devices.

`DeviceItem` objects are nested children of a `Device` object. An object of the type `DeviceItem` can be a rack or an inserted module.

- The first child item of a device is of type "rack". The `PositionNumber` of a rack starts with 0. If there are multiple racks, they are consecutively numbered (1, 2, 3, ...).
  There are no restrictions about the ordering in the AML file within one hierarchy level.
- All further children of the type "rack" are modules.

The CAx data export supports the following types of device items specified via the AML type identifier:

- Physical modules
- GSD/GSDML modules

Attributes

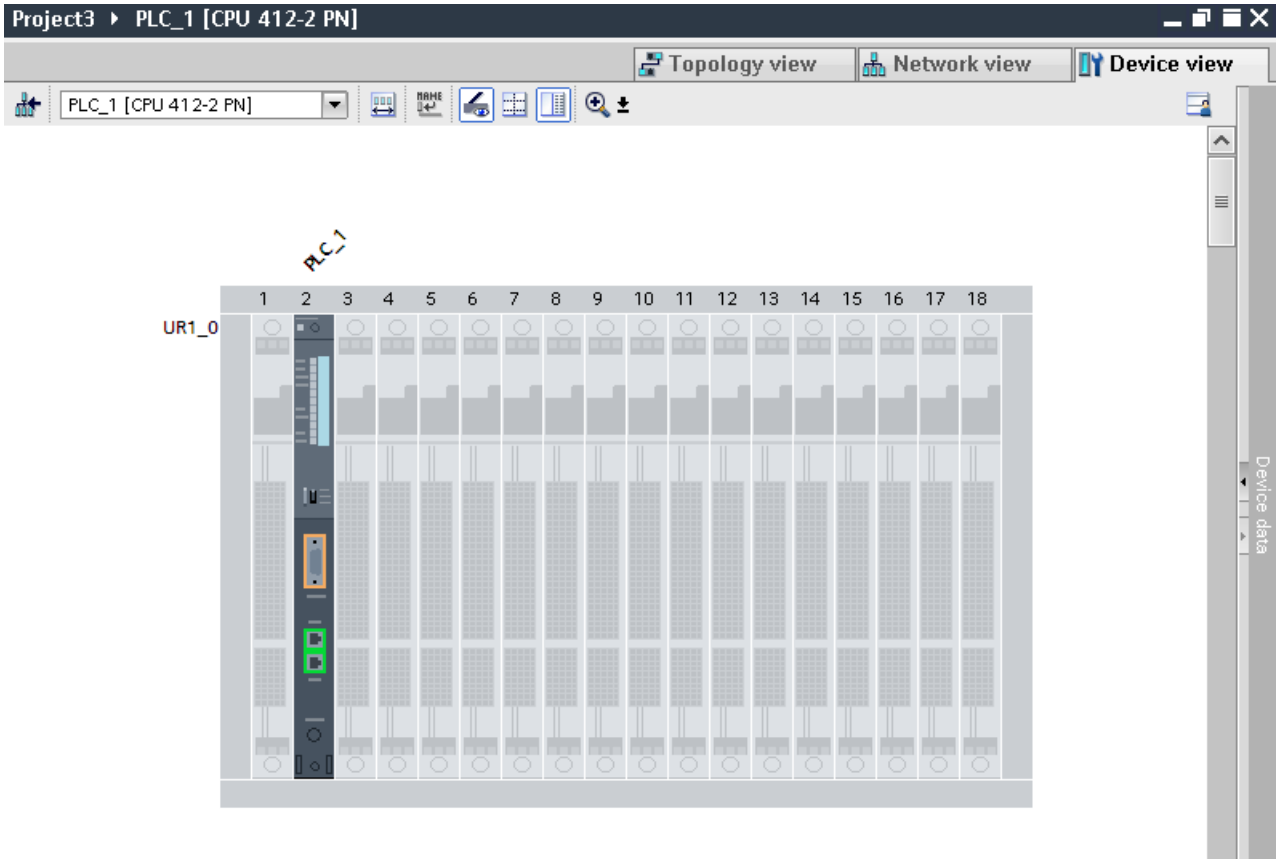
The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory Export-only for "BuiltIn" = TRUE	
TypeName	Export-only for "BuiltIn" = FALSE	
DeviceItemType	Export-only	Only for PLC (central devices) and device items (physical racks, modules, HeadModule). Optional during import, but any device item except Base units with DeviceItemType as accessory shall be silently ignored.
PositionNumber	Mandatory, Optional for certain device items	For certain device items, Position number is not relevant for exchange with external tools like ECAD Systems. For such device items, Position number will not be mandatory in AML file and will be assigned an appropriate value internally by import operation.
BuiltIn	Optional	Default: FALSE
TypeIdentifier	Mandatory for "BuiltIn" = FALSE Ignored for "BuiltIn" = TRUE	For integrated built-in device items, this attribute will be exported with its pluggable parent type identifier information.and have no relevance during import and hence it is optional.For non-integrated built-in device items, this attribute is optional
FirmwareVersion	Optional, mandatory if the object supports firmware versions	
PlantDesignation IEC	Optional	Default: ""
LocationIdentifier IEC	Optional	Default: ""
Comment	Optional for "BuiltIn" = FALSE	Default: ""
ProductDesignation IEC	Mandatory if the object supports this attribute in TIA Portal and is non empty	Import is not supported, if the length of ProductDesignation IEC is greater than 54 characters. Export/ Import under device items as a part of AR APC V1.1.0 support for TIA Portal V16
InstallationDate	Mandatory if the object supports this attribute in TIA Portal	Export/ Import under device items as a part of AR APC V1.1.0 support for TIA Portal V16
Address	Optional	"Address" has nested attributes

The following table shows the nested attributes of the "Address" attribute of the object "DeviceItem":

Attributes for "Address"	Handling	Comment
StartAddress	Mandatory	
Length	Export-only	Export/Import of address with Length = 0 is not supported.
IoType	Mandatory	Input or Output

Example: Exported Configuration



AML structure of the export file

The following structure example depicts the export of "UR1_0" and the module "PLC_1".

```
<InternalElement ID="7624ed42-3db7-4ba5-8726-e719d7b09969" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 station</Value>
  </Attribute>
  <InternalElement ID="1ff247e6-6b94-42a2-bcd5-d673fc6e9ba5" Name="UR1_0">
    <Attribute Name="TypeName" AttributeDataType="xs:string">
      <Value>UR1</Value>
    </Attribute>
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>False</Value>
    </Attribute>
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>OrderNumber:6ES7 400-1TA01-0AA0</Value>
    </Attribute>
    <Attribute Name="Comment" AttributeDataType="xs:string">
      <Value>S7 400 rack</Value>
    </Attribute>
    <InternalElement ID="202421bf-a4b9-4399-a563-9f154f0eabab" Name="PLC_1">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>CPU 412-2 PN</Value>
      </Attribute>
      <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
        <Value>CPU</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>2</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>OrderNumber:6ES7 412-2EK06-0AB0</Value>
      </Attribute>
      <Attribute Name="Comment" AttributeDataType="xs:string">
        <Value>S7 400 plc</Value>
      </Attribute>
      <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
        <Value>V6.0</Value>
      </Attribute>
      <Attribute Name="ProductDesignation IEC" AttributeDataType="xs:string">
          <Value>Additional Information</Value>
      </Attribute>
      <Attribute Name="InstallationDate" AttributeDataType="xs:dateTime">
          <Value>2018-11-26T05:09:02.803Z</Value>
      </Attribute>
      <Attribute Name="PlantDesignation IEC" AttributeDataType="xs:string">
        <Value>PD</Value>
      </Attribute>
      <Attribute Name="LocationIdentifier IEC" AttributeDataType="xs:string">
        <Value>LI</Value>
      </Attribute>
      <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/AutomationProject" />
</InternalElement>
```

See also

Export/Import of GSD/GSDML based devices and device items (Page 1578)

Structure of the CAx data for importing/exporting (Page 1435)

AML type identifiers (Page 1439)

6.5.29 Import of device item objects

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- The PLC is offline.

Application

The import of device item objets is only applicable for PLC devices.

`DeviceItem` objects are nested children of a `Device` object. An object of the type `DeviceItem` can be a rack or an inserted module.

- The first child item of a device has to be of type rack. The `PositionNumber` of a rack starts with 0. If there are multiple racks, they are consecutively numbered (1, 2, 3, ...).
  There are no restrictions about the ordering in the AML file within one hierarchy level.
- All further children of the type rack are modules.

The CAx data imporet supports the following types of device items specified via the AML type identifier:

- Physical modules
- GSD/GSDML modules
- Generic modules

If you only know the identification (`TypeIdentifier`) of a head module or a PLC and not of the respective rack and device, you can import a generic rack.

Example: `TypeIdentifier` = `System:Rack.Generic`

For generic rack replacement, the following elements have to be present in the rack described in the AML file:

- Central devices: PLC
- Decentral devices: Head module

A generic rack type derives from the Device type. Therefore, an AML file that is imported to TIA Portal can use this rack's type identifier:

In this case the TIA Portal determines the type identifier for the rack.

If racks and modules are generic, the attribute `BuiltIn` defines the kind of rack or module:

- physical: `BuiltIn = True`
- generic: `BuiltIn = False`

Restrictions

While importing, the attribute `DeviceItemType` has no relevance and hence it is optional.

Note

Attribute "FirmwareVersion"

If no `FirmwareVersion` is specified in the import file CAx import uses the latest firmware version which is available in the TIA Portal

If the `FirmwareVersion` attribute exists in the import file with an empty value, the device item import fails and an error message will be logged.

Example: Importing a generic device

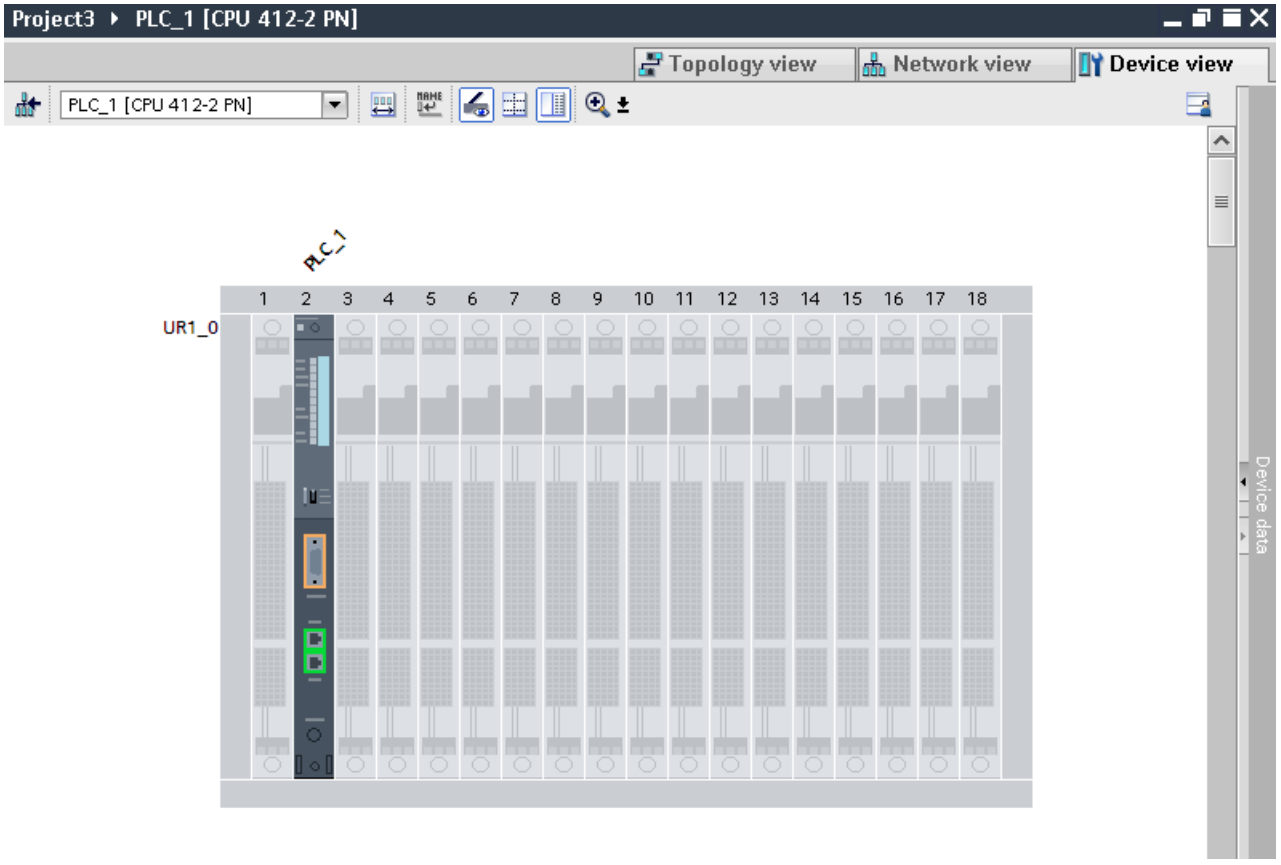
The following structure example depicts the import of the generic rack "Rack_1".

```
...
<InternalElement ID="6563466e-2de9-42ca-951d-eb8f2545958d" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <InternalElement ID="96930368-14ec-43e2-b9b7-c1fefc4b0534" Name="UR1_0">
    <Attribute Name="TypeName" AttributeDataType="xs:string">
      <Value>UR1</Value>
    </Attribute>
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>False</Value>
    </Attribute>
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Rack.Generic</Value>
    </Attribute>
    <InternalElement ID="a1de449e-4f89-45af-8bbc-f77c28bccd04" Name="PLC_1">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>CPU 412-2 PN</Value>
      </Attribute>
      <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
        <Value>CPU</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>2</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>OrderNumber:6ES7 412-2EK06-0AB0</Value>
      </Attribute>
      <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
        <Value>V6.0</Value>
      </Attribute>

      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
...
```

Imported Configuration

The following figure shows the imported configuration in the TIA Portal user interface:



See also

Structure of the CAx data for importing/exporting (Page 1435)

AML type identifiers (Page 1439)

6.5.30 Export/Import of GSD/GSDML based devices and device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

The CAx Import/Export of GSD/GSDML based devices and device items is similar to the import/export of standard devices.

For GSD/GSDML based devices and device items the exportable attributes differ, e. g. for GSD/GSDML the attribute `Label` exists.

Generic import of devices and racks are possible. For the import, you use the same identifier as for standard devices:

- Import of a generic device: `TypeIdentifier = System:Device.Generic`
- Import of a generic rack: `TypeIdentifier = System:Rack.Generic`

If devices are generic, the attribute `BuiltIn` defines the kind:

- physical: `BuiltIn = True`
- generic: `BuiltIn = False`

Attributes for a device

The following table shows the related attributes of device for CAx import and export files:

Attribute	Handling for attribute	Comment
Name	Mandatory for export and import	
TypeIdentifier	Mandatory for export and optional for import starting from AR APC V1.1	
Comment	Optional for import	

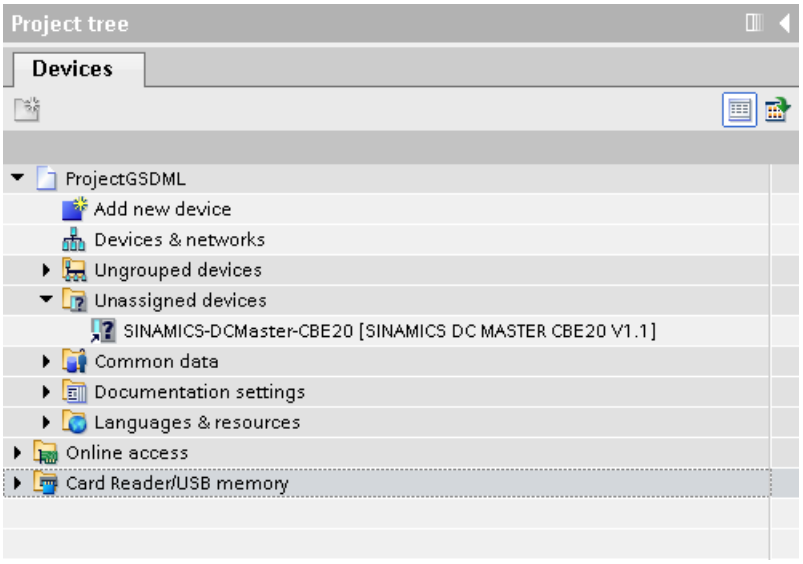
Attributes for a device item

The following table shows the related attributes of a device item for CAx import and export files:

Attribute	Handling for attribute BuiltIn = FALSE Generic device items	Handling for attribute BuiltIn = TRUE Physical device items	Comment
Name	Mandatory	Export-only	
TypeName	Export-only	Not applicable	
DeviceItem-Type	Export-only	Export-only	Only for PLC (central devices) and Head-Module (decentral devices) device items Optional during import, but any device item except Base units with DeviceItem-Type as accessory shall be ignored.
PositionNum-ber	Mandatory	Mandatory for export Exceptional cases: Device item type interface: Optional for import Device item type port: Optional	
BuiltIn	Optional		Default: FALSE

Attribute	Handling for attribute BuiltIn = FALSE Generic device items	Handling for attribute BuiltIn = TRUE Physical device items	Comment
TypeIdentifier	Mandatory for "BuiltIn" = FALSE	Ignored for "BuiltIn" = TRUE	For integrated built-in device items, this attribute will be exported with its pluggable parent type identifier information. This attribute has no relevance during import, hence it is optional. For non-integrated built-in device items, this attribute has no relevance.
Comment	Optional	-	
Label	-	- Device item type interface: Mandatory Device item type port: Mandatory	

Example: Exported GSD/GSDML device



AML structure of the export file

The following figure shows the structure of the exported AML file.

```
...
<InternalElement ID="9ae02cde-dfb4-4d43-a649-68b9ede7fc3d" Name="Ungrouped devices">
  <InternalElement ID="12d4ce0f-346d-4bfa-b139-c9d0db64c794" Name="GSD device_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/D</Value>
    </Attribute>
    <InternalElement ID="ccb1cb62-67b2-4b8c-951f-10c7ffb4d787" Name="Rack">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>Rack</Value>
      </Attribute>
      ...
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/R/IDD_14</Value>
      </Attribute>
      <InternalElement ID="74f25b5c-0c09-46d0-9011-f341a3e98a0d" Name="SINAMICS-DCMaster-CBE20">
        <Attribute Name="TypeName" AttributeDataType="xs:string">
          <Value>SINAMICS DC MASTER CBE20 V1.1</Value>
        </Attribute>
        <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
          <Value>HeadModule</Value>
        </Attribute>
        <Attribute Name="PositionNumber" AttributeDataType="xs:int">
          <Value>0</Value>
        </Attribute>
        <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
          <Value>False</Value>
        </Attribute>
        <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
          <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/DAP/IDD_14</Value>
        </Attribute>
        <InternalElement ID="94f34bb9-fe47-4904-b8a1-62e8fb6b1b74"
            Name="SINAMICS DC MASTER CBE20 V1.1">
          <Attribute Name="PositionNumber" AttributeDataType="xs:int">
            <Value>0</Value>
          </Attribute>
          <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
            <Value>True</Value>
          </Attribute>
          <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
        </InternalElement>
        ...
        <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
      </InternalElement>
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
</InternalElement>
...
```

Generic & Non generic rack with & without TypeIdentifier

The CAx import shall be able to handle devices with no type identifier information or generic type identifiers i.e.'System:Device.Generic' and racks with generic type identifiers i.e. 'System:Rack.Generic'.

While importing it shall be possible that AML file contains certain types of devices with no type identifier or generic type identifier i.e 'System:Device.Generic' and rack device items with generic type identifier i.e, 'System:Rack.Generic'. But CAx import shall handle these as well to create proper devices and rack device items.

Following devices support generic device , device with no type identifier and generic rack substitution:

- GSD and GSDML devices - All the GSD/GSDML devices with GSD/GSDML racks.
- Devices based on MDD (Non-GSD/GSDML devices) - The devices with system rack type identifiers.

CAx export has no relevance with generic rack handling. CAx always exports non generic rack type identifiers.

For Generic device or device having no type identifier and Generic Rack, type identifier replacement , it is mandatory that the head module (in case of decentral devices) or PLC (in case of central devices) has to be present in the rack described in the AML file, otherwise the device with no type identifiers or generic device and generic rack type identifier substitution shall fail.

The following XML structure shows a GSDML device configuration with device (with Type Identifier) and rack (with TypeIdentifier):

```
<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD device_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.32-SIEMENS-SINAMICS_DCMASTER-20160531.XML/D</Value>
</Attribute>
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.32-SIEMENS-SINAMICS_DCMASTER-20160531.XML/R/IDD_14</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
```

The following XML structure shows a GSDML device configuration with device (no type identifier) and rack (generic type identifier):

```
<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD device_1">
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Rack.Generic</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
```

The following XML structure shows a GSDML device configuration with device (generic type identifier) and rack (generic type identifier):

```
<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD device_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.Generic</Value>
</Attribute>
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Rack.Generic</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.32-SIEMENS-SINAMICS_DCMASTER-20160531.XML/DAP/IDD_14</Value>
</Attribute>
```

See also

AML type identifiers (Page 1439)

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.31 Export/Import device configuration with virtual interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

In TIA Portal, ports are used for inter-device communication and found under interfaces. But there are certain kind of devices where the ports are found directly under the device items which are not interfaces. This is not in accordance with the AML standards where ports are always looked for under interfaces.

CAx shall use an imaginary interface referred to Virtual interface to export and import device configuration in case ports are found directly under non-interface device items.

Export of AML file

The below example shows an AML file having virtual interface:

```
<InternalElement ID="822622a0-056a-494c-a802-2463c5e1b47d" Name="SCALANCE interface_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="5a604a57-bc2d-4763-8df0-7d20000faf1b" Name="VirtualInterface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="3c4862a1-b8ed-4610-88f3-29dc8328e748" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="220dd49d-8d23-44b1-bdc1-878516540313" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="99c6253b-c546-4720-af54-92db926b8231"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

The virtual interface shall be exported with the following attributes till before TIA Portal v16 :

- Name as ScalanceInterface_1
- Label as Switch

From TIA Portal v16 the virtual interface shall be exported with the below attributes

- Name as VirtualInterface_1
- Label as X1
- Type as Ethernet

Import

CAx shall support import of virtual interfaces. In this case, it is expected to process the ports under the Virtual interface in the AML file under the right parent in the TIA Portal. Here any interface with Label as switch is to be considered as virtual interface. But with TIA Portal v16, Label is nor more identifier as it is changed to X1, which is just like a real interface.

Type attribute although is set to "Ethernet" for virtual interface and it is optional. So with TIA Portal v16 all interface elements in AML file not labelled to "Switch" shall be treated in a general way, and whenever encountered in AML file CAx shall try to fetch from TIA Portal.

In case of Virtual interface, CAx shall fail to find it but process the ports inside it. But in case of real interface if CAx doesn't find it, the ports inside it shall not be processed

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.32 Export/Import of subnets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

AML structure

Subnets describe a physical network especially which devices are connected to the same network of type PROFIBUS, PROFINET, MPI or ASI.

The link between a network and the device items are modeled as a reference to the network object. There is no reference from the network object to the attached device items. The network parameters are stored in the network object. The parameters concerning a network

interface of a given device item, attached to a network, are stored in a net node object in that device item. The communication is often regulated using “channels”, “ports” and “interfaces”.

Subnets are exported as internal elements of the role class "Subnet" in the instance hierarchy in the AML file.

A subnet has the following related elements in the AML structure:

- Internal element of the role class "Node"
  Defines the interface on a device item.
- <InternalLink>
  Defines the connected partners of the subnet. <InternalLink> tags name is unique and is always added under the project's internal element in the AML file.

```
    ....
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
    <InternalLink Name="Link To Port_1"
      RefPartnerSideA="1e3e4c5b-04c1-4d2c-9aee-cad53cc92dba:CommunicationPortInterface"
      RefPartnerSideB="d45aa36a-a7f2-4862-a266-d6727b9cfd75:CommunicationPortInterface" />
    <InternalLink Name="Link To Subnet_1"
      RefPartnerSideA="beb4eb8e-1a45-45ce-a703-1acfac73e5f3:LogicalEndPoint_Node"
      RefPartnerSideB="1062a384-d3ca-4183-9ac2-0934a5ab7286:LogicalEndPoint_Subnet" />
    <InternalLink Name="Link To Subnet_2"
      RefPartnerSideA="a3e85aed-580a-45c8-943e-da7de8280b7c:LogicalEndPoint_Node"
      RefPartnerSideB="1062a384-d3ca-4183-9ac2-0934a5ab7286:LogicalEndPoint_Subnet" />
  </InternalElement>
 </InstanceHierarchy>
</CAEXFile>
```

- <ExternalInterface>
  Represents in node and subnet internal elements that nodes and subnets are connected. If the nodes or subnets are not connected then the <ExternalInterface> element for node and subnet do not exist.

```
...
<InternalElement ID="1062a384-d3ca-4183-9ac2-0934a5ab7286" Name="PN/IE_1">
  <Attribute Name="Type" AttributeDataType="xs:string">
    <Value>Ethernet</Value>
  </Attribute>
  <ExternalInterface ID="55ecf2cb-e72a-4974-8ed0-1d4e1ade3509"
      Name="LogicalEndPoint_Subnet"
      RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Subnet" />
</InternalElement>
...
```

Application

In TIA Portal V16,CAx import supports Ethernet and Profinet and export will always be Ethernet for Subnet Type and Node attributes.

The CAx Import/Export supports the following types of subnets:

- Ethernet
- PROFIBUS

- MPI
- ASi

Attributes of a "Subnet" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	
Type	Mandatory	Ethernet or PROFIBUS or MPI or ASi

Attributes of "CommunicationInterface" elements

The following table shows the related attributes of the objects for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	No relevance for "fixed" device items.
Label	Mandatory	Label may be missing if "BuiltIn" = TRUE and "PositionNumber" are specified for the related "DeviceItem" object.
TypeIdentifier	Mandatory	For integrated built-in device items, this attribute shall be exported with its pluggable parent type identifier information. And during import this attribute has no relevance. For non-integrated built-in device items, this attribute has no relevance.
FirmwareVersion	Mandatory	
TypeName	Export-only	No relevance for "BuiltIn" device items.
DeviceItemType	Export-only	Only for for CPU and HeadModule This attribute is optional during import but any device item except Base units with DeviceItemType as Accessory shall be silently ignored.
PositionNumber	Mandatory	No relevance for the import of "BuiltIn" device items.
BuiltIn	Mandatory for export Optional for import	No relevance for the import of "Non-BuiltIn" device items. False by default for import.
Comment	Optional	Not applicable for "BuiltIn" device items.

Attributes of "CommunicationPort" elements

The following table shows the related attributes of the objects for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	No relevance for "BuiltIn" device items.
Label	Mandatory	The Label attribute has no relevance for non built-in port device items and hence shall not be exported. From TIA Portal V19 onwards, Label value shall be without space (For example P1R) while exporting port configuration. While importing Port label value, comparison of label value shall be case invariant (For ex:ample p1 r, P1R, P1, p1 etc.,). For Ports enabled for ring topology, indicated with suffix "R", import shall be treated "equivalent" to Ports having label values without suffix. For example: The following port labels are treated as equal: • P1 equals P1 • P1 equals P1R • P1R equals P1 • P1R equals P1R
TypeIdentifier	Mandatory	This attribute is mandatory for export and import of non built-in device item. For example device items with BuiltIn = False For integrated built-in device items, this attribute shall be exported with its pluggable parent type identifier information. And during import this attribute has no relevance. For non-integrated built-in device items, this attribute has no relevance
FirmwareVersion	Mandatory	This is a mandatory attribute for both export and import if and only if the device item is non built-in and it supports firmware version. This attribute is not applicable if the device item is built in.
TypeName	Mandatory for Export Optional for import	No relevance for "BuiltIn" device items.
PositionNumber	Mandatory	This attribute is mandatory for export for all types of port device items. During import, the attribute is mandatory for non built-in port device items, but has no relevance for built-in port device items
BuiltIn	Mandatory for export Optional for import	No relevance for the import of "Non-BuiltIn" device items. False by default for import.
Comment	Optional	Not applicable for "BuiltIn" device items.

Attributes of a "Node" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Export-only	MPI, PROFIBUS, PROFINET
Type	Export-only	Ethernet or PROFIBUS or MPI or ASi
NetworkAddress	Mandatory	
SubnetMask	Optional	PROFINET For import, default value is retained if no value is set.

Attribute	Handling	Comment
RouterAddress	Optional	PROFINET For import, default value is retained if no value is set.
DhcpClientId	Optional	PROFINET For import, default value is retained if no value is set.
IpProtocolSelection	Optional	PROFINET For import, default value is retained if no value is set. Values: Project, Dhcp, UserProgram, OtherPath, Address Tailoring

For Profinet node, the availability of attributes which will differ again based on certain pre conditions. For ex:

- If in TIA Portal UI Ip protocol selected is "Set IP address in the project", then "NetworkAddress" and "SubnetMask" will be available.
- The attribute "RouterAddress" shall be available for export/import if Ip protocol selected is "Set IP address in the project" and "Use router" is selected in UI.
- The attribute "DhcpClientId" shall be available for export/import only if the attribute 'IpProtocolSelection' is 'Dhcp'.

CAx Import special case where 'Type' attribute necessary - Profibus subnet connection is done by changing interface type MPI to PROFIBUS in UI.

- In TIA portal for some device items, by default interface type is Mpi. But it is required to be changed to Profibus to make Profibus subnet connection. In this case, Type should be specified with 'Profibus' value in AML file to indicate Type conversion while importing.

Note

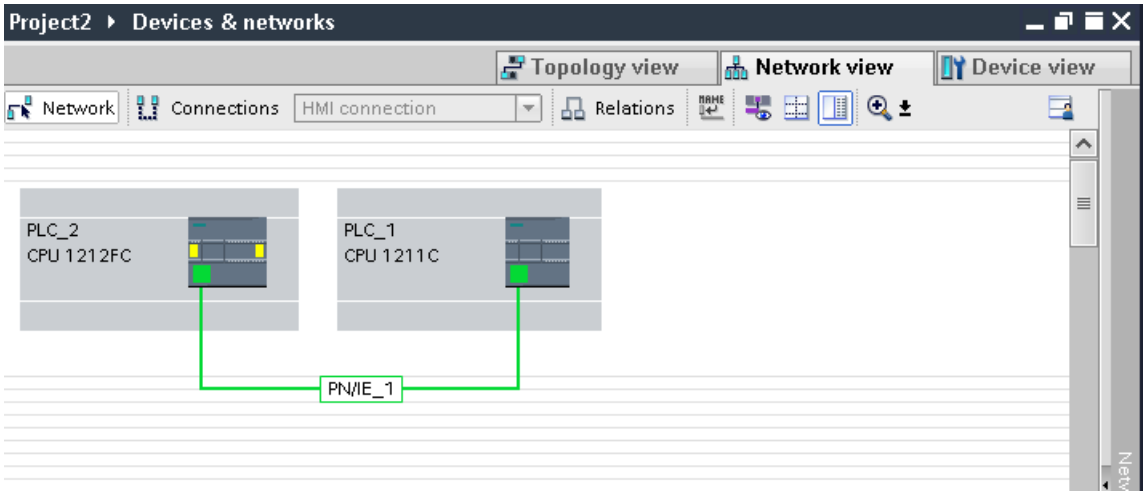
- CAx import will support Ethernet and Profinet for Subnet Type and Node Type attribute.
- CAx import of an AML file with 'case invariant' string values for the attributes (For ex: Profinet, PROFINET, pROFINET, ProFINET etc.,) should be supported.
- CAx export will always be Ethernet.

Attributes of a "Channel" element"

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Type	Mandatory	Digital or Analog
IoType	Mandatory	Input or Output
Number	Mandatory	
Length	Export-only	

Example: Exported subnet



AML structure

The following figures show the structure of the exported AML file:

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="e9d2bedb-f8c1-4148-acda-c3c68836c7dd" Name="Project2">
    ...
      <InternalElement ID="1062a384-d3ca-4183-9ac2-0934a5ab7286" Name="PN/IE_1">
      <Attribute Name="Type" AttributeDataType="xs:string">
        <Value>Ethernet</Value>
      </Attribute>
      <ExternalInterface ID="55ecf2cb-e72a-4974-8ed0-1d4e1ade3509"
          Name="LogicalEndPoint_Subnet"
              RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
      <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Subnet" />
    </InternalElement>
    <InternalElement ID="b011dbb1-efa4-46c0-a26f-f9bd047cda4f" Name="S7-1200 station_1">
          ...
          <InternalElement ID="d006e41b-05ff-44ab-baab-fca15f99e86c" Name="PROFINET interface_1">
            ...
            <InternalElement ID="beb4eb8e-1a45-45ce-a703-1acfac73e5f3" Name="E1">
              ...
              <ExternalInterface ID="a365b498-20cc-4e0b-99ca-5c5257632b96"
                  Name="LogicalEndPoint_Node" RefBaseClassPath=
                      "CommunicationInterfaceClassLib/LogicalEndPoint" />
              <SupportedRoleClass RefRoleClassPath=
                  "AutomationProjectConfigurationRoleClassLib/Node" />
            </InternalElement>
            <InternalElement ID="d45aa36a-a7f2-4862-a266-d6727b9cfd75" Name="Port_1">
              ...
              <ExternalInterface ID="32c6ba4a-b01f-4678-b721-ea284779e96c"
                  Name="CommunicationPortInterface"
                  RefBaseClassPath=
                  "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
              <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
            </InternalElement>
            <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
          </InternalElement>
          <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
        </InternalElement>
        <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
      </InternalElement>
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
  ...
```

```
...
<InternalElement ID="7cf0ea2b-b66f-4ad4-8a03-5a8691cbe04d" Name="PLC_2">
  ...
  <InternalElement ID="b287020d-667b-483d-a8e0-c5466ac2f5c3" Name="PROFINET interface_1">
    <Attribute Name="Label" AttributeDataType="xs:string">
      <Value>X1</...
    <InternalElement ID="a3e85aed-580a-45c8-943e-da7de8280b7c" Name="E1">
      <Attribute Name="Type" AttributeDataType="xs:string">
        <Value>Ethernet</Value>
      </Attribute>
      <Attribute Name="NetworkAddress" AttributeDataType="xs:string">
        <Value>192.168.0.2</Value>
      </Attribute>
      <Attribute Name="SubnetMask" AttributeDataType="xs:string">
        <Value>255.255.255.0</Value>
      </Attribute>
      <Attribute Name="DeviceNumber" AttributeDataType="xs:string">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
        <Value>Project</Value>
      </Attribute>
      <ExternalInterface ID="6ae8eb93-09d3-4f8c-b529-a12148c71bf4"
          Name="LogicalEndPoint_Node"
          RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
      <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
    </InternalElement>
    <InternalElement ID="1e3e4c5b-04c1-4d2c-9aee-cad53cc92dba" Name="Port_1">
      ...
      <ExternalInterface ID="1f5b2a3d-fcd1-460a-b846-30dadc8726d1"
          Name="CommunicationPortInterface"
          RefBaseClassPath=
          "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
      <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
...
```

Extended role for Profinet/Ethernet node

As part of latest recommendation by AR APC 1.2, any Profinet/Ethernet node shall be exchanged with one additional role 'AutomationProjectConfigurationRoleClassLib/NodeEthernet' along with the existing role 'AutomationProjectConfigurationRoleClassLib/Node' in case extended attributes are supported.

The Profinet/Ethernet node shall be exported with additional role in case it supports extended attributes like SubnetMask, RouterAddress, DhcpClientId, IpProtocolSelection. And it shall be possible to import node with or without extended role also.

Below XML snippet depicts the AML file with Node "Ethernet" with extended role.

```
<InternalElement ID="9c4393a9-44d5-49b4-9314-02eb0f94b6c0" Name="IE1">
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="RouterAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<ExternalInterface ID="eea59d3c-3bc4-4d0d-9815-46b4b347369d" Name="LogicalEndPoint_Node"
RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationEthernetRoleClassLib/
NodeEthernet" />
</InternalElement>
```

See also

Structure of the CAx data for importing/exporting (Page 1435)

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.33 Export/Import of IO-systems

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

AML structure

IO-system are represented in the AML structure as `<InternalElement>`.

IO-system of as master or IO controller are added under the element `<CommunicationInterface>` of an interface device item.

```
...
<InternalElement ID="[Communication Interface UniqueID]"
      Name="[Communication Interface Name]">
...
  <!--Node-->
  <InternalElement ID="[Node UniqueID]" Name="[Node Name]">
    ...
    <ExternalInterface ID="[External Interface UniqueID]"
      Name="LogicalEndPoint_Node"
      RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
    <SupportedRoleClass
      RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
  </InternalElement>
  <!--IoSystem-->
  <InternalElement ID="[IoSystem UniqueID]" Name="[IoSystem Name]">
   <Attribute Name="Number" AttributeDataType="xs:integer">
    <Value>[IoSystem Number]</Value>
    </Attribute>
    <ExternalInterface ID="[External Interface UniqueID]"
      Name="LogicalEndPoint_Interface"
      RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
    <SupportedRoleClass
      RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/IoSystem" />
  </InternalElement>
  <SupportedRoleClass
      RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
```

Connected IO-system as slave or IO-device are added as <ExternalInterface> elements under the element <CommunicationInterface> of an interface device item.

```
<InternalElement ID="[Communication Interface UniqueID]"
    Name="[Communication Interface Name]">
 ...
  <ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Interface"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
  <!--Node-->
  <InternalElement ID="[Node UniqueID]" Name="[Node Name]">
    <ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Node"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
    <SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
  </InternalElement>
  <SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
```

The connected partners of the IO-systems are represented as <InternalLink> elements. <InternalLink> tags are added under common parent of an IO-system and connected slave device item, e. g. Project, DeviceFolder, DeviceItem.

<InternalLink> tags name is unique across the common parent.

Attributes of an "IO-system" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	The IO-system name. If empty string is imported, the IO-system is created with the default name.
Number	Optional	If not specified for import, the default value is applied.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.34 Export/Import of multilingual comments

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

The CAx data exchange exports and imports comments and multilingual comments of the following hardware objects:

- Devices (Device)
- Modules (DeviceItems)
- Tags (Tag)

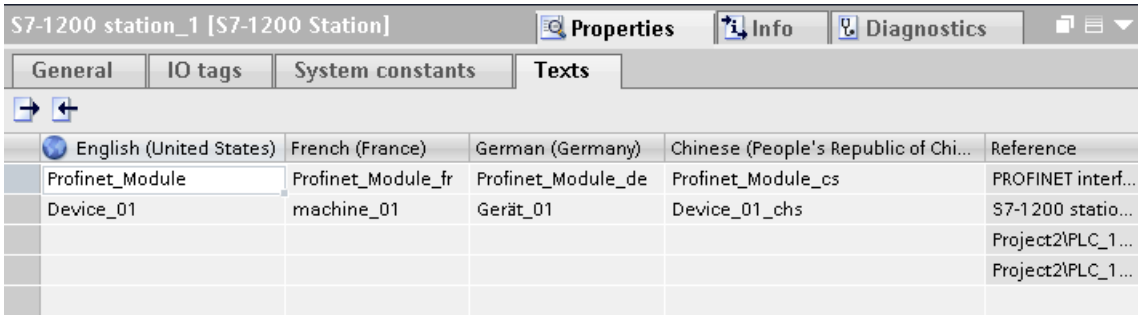
The import/export of multilingual comments comprises all TIA Portal languages.

Restrictions

- Export
  - Only if a comment exists, a "Comment" attribute is exported to the AML file.
- Import
  - The "Comment" attribute is optional.
  - For virtual device items, no comments can be imported.

Example: Exported configuration with multilingual comments

The following figure shows the configuration of a SIMATIC S7 1500 (Device) with PLC_1 (DeviceItems). For both objects, comments are set in English, French, German and Chinese.



AML structure

After the export of this configurations, the multilingual comments are generated as nested attributes of the device, device item or tag.

- The parent attribute "Comment" shall have the value used in default language.
- A child attribute exists for every foreign-language comment.

```
...
<Attribute Name="Comment" AttributeDataType="xs:string">
  <Value>English</Value>
  <Attribute Name="aml-lang=en-US" AttributeDataType="xs:string">
    <Value>English</Value>
  </Attribute>
  <Attribute Name="aml-lang=de-DE" AttributeDataType="xs:string">
    <Value>Deutsch</Value>
  </Attribute>
  <Attribute Name="aml-lang=zh-HK" AttributeDataType="xs:string">
    <Value>Chinese</Value>
  </Attribute>
  ...
  ...
</Attribute>
...
```

See also

Structure of the CAx data for importing/exporting (Page 1435)

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.35 Export/Import ET200 SP/ET200 AL devices with extension rack connections

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is opening
  See Opening a Project (Page 128)

Application

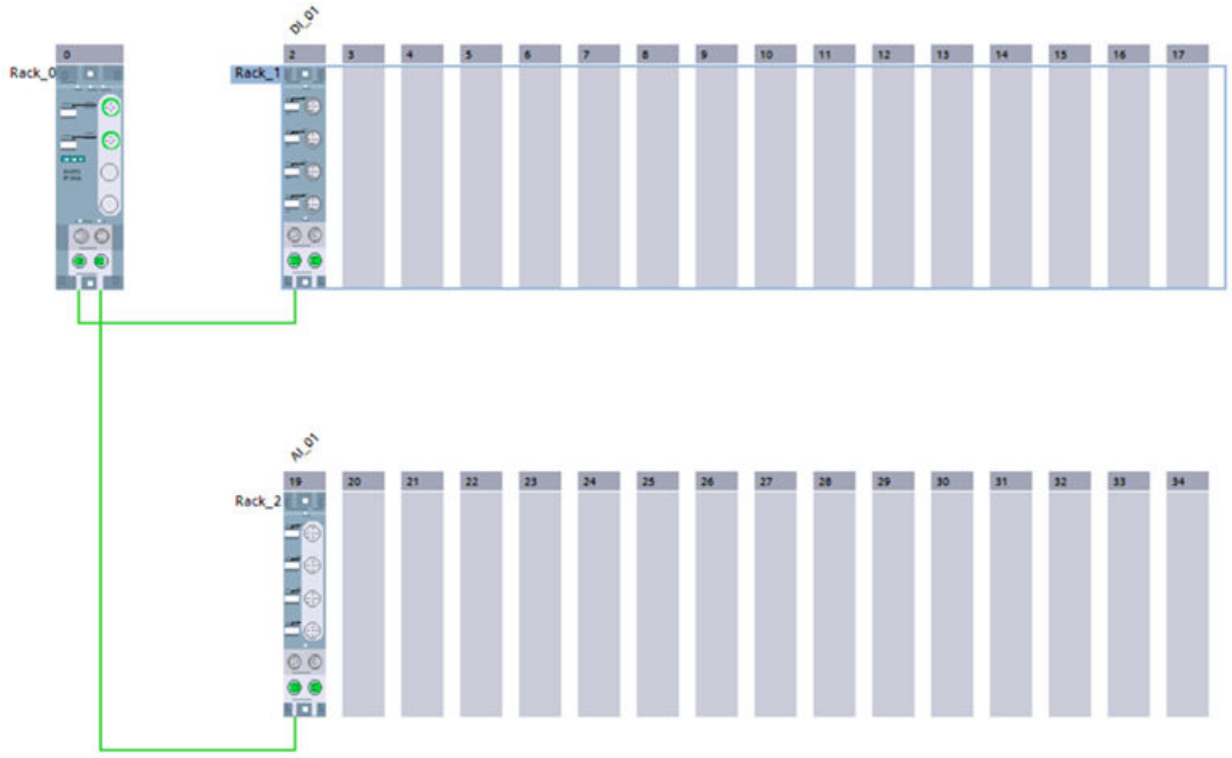
It shall be possible to have a device with multiple racks having extension rack connections exported in to AML file and import it to get same device with extension rack connections created back in TIA Portal project.

In TIA Portal, extension rack connections between multiple racks are modeled under DeviceItem objects (ET-Con_1, ET-Con_2 and ET-Connection Receiver) directly. However as per AR APC recommendation, these connections to be modeled as Port-to-Port connection under a CommunicationPort.

So, to make it in line with the recommendation, a dummy CommunicationInterface having dummy CommunicationPort objects shall be added under respective DeviceItem objects.

Device configurations with multiple extension rack connections

The following configuration shows a ET-200AL device with multiple racks having extension rack connections in TIA Portal.



AML Export

The AML file that shall be generated during export for the above configuration is depicted below.

```
<?xml version="1.0" encoding="utf-8"?><CAEXFile FileName="ET200AL_with_ExtensionRacks.aml"
SchemaVersion="2.15" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
...
<InternalElement ID="2b90fa1e-df69-437d-8a77-eaa455cdfaba" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ET-Con</Value>
</Attribute>
...
<InternalElement ID="32758129-15db-43c7-9deb-53613852b208" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X30</Value></Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="c326d6b8-cc1a-444b-8a06-5c63a226a456"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalElement ID="ab4e0e97-8e57-4218-b55c-0e4fe89a9587" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ET-Con</Value>
</Attribute>
...
<InternalElement ID="6bb61ad0-13ac-455b-9a90-6cbaddeaabd4" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X31</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="90c95545-2e43-4701-8fd0-9e87ef0bd412"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<InternalElement ID="54119ad3-34b6-4e63-bc18-bee23312900a" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
```

```
<Value>ET-Con</Value>
</Attribute>
...
<InternalElement ID="ad638a2a-538e-4840-9d99-7712522431ba" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X30</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="6a517987-ae22-4446-b361-91aea2c768be"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalLink Name="Link To Port_1"
RefPartnerSideA="32758129-15db-43c7-9deb-53613852b208:CommunicationPortInterface"
RefPartnerSideB="ad638a2a-538e-4840-9d99-7712522431ba:CommunicationPortInterface" />
<InternalLink Name="Link To Port_2"
RefPartnerSideA="6bb61ad0-13ac-455b-9a90-6cbaddeaabd4:CommunicationPortInterface"
RefPartnerSideB="cac8af19-67bc-41aa-a3f0-7149af8f67d4:CommunicationPortInterface" />
```

Note

The extension rack interface shall be exported only when extension rack connections are present. Also, the number of dummy ports added under extension ET-Con dummy interface is based on the ports participating in extension rack connection at module level.

Extension rack connection

The XML representation of extension rack connections between multiple racks shall be done using format shown below.

- **ExternalInterface**-<ExternalInterface> internal element shall be added under <CommunicationPort> internal element which participates in the connection

```
<InternalElement ID="[IM Module Unique ID]" Name="[IM Module Name]">
<InternalElement ID="[Dummy Interface Unique ID]" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ET-Con</Value>
</Attribute>
...
<InternalElement ID="[Dummy Port Unique ID]" Name="[IM Module Sender/Receiver Name]">
...
<ExternalInterface ID="[External Interface Unique ID]" Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="[Dummy Port Unique ID]" Name="[IM Module Sender/Receiver Name]">
...
<ExternalInterface ID="[External Interface Unique ID]" Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

Internal link- Extension racks connections are represented using <InternalLink> tags. <InternalLink> tags shall be added under common parent of multiple racks (i.e., Device). Internal link name shall be unique across the common parent.

```
<InternalLink Name="Link To [Internal link Name]" RefPartnerSideA="[Communication Port
UniqueID]:[Communication Port External Interface Name]" RefPartnerSideB="[Communication
Port UniqueID]:[Communication Port External Interface Name]" />
```

Import AML

It shall be possible to import extension rack connections details from an AML file which is generated out of above-mentioned export. However, it shall also be possible to import AML files which are created in previous versions of TIA Portal.

Note

1. The export hierarchy change behavior shall be applicable only in version V16 onwards. Older version TIA portal shall have same behavior as previous.
2. The hierarchy in AML file shall not influence/affect the TIA Portal internal hierarchy after import.
3. This hierarchy change/transformation behavior is applicable for ET200 AL and ET200 SP with extension rack connections.
4. The AML file with "case invariant" string values for the Type attribute (For ex: ET-Con, et-con, ET-con etc.,.) shall also be imported without any failure.

Extension racks with Side by Side Connections

The following configuration shows as extension rack with modules which are connected side by side. In below mentioned configuration, modules DI_01, DI_02, DI_03, AQ_01 and AQ_02 connected to their subsequent modules in TIA Portal. However, in TIA Portal connection between BusAdapter-Sender and DI_01 modules are modeled as Extension rack connection and other module connections not modeled.

The AML file that shall be generated during export for the above configuration is depicted below.

```
<?xml version="1.0" encoding="utf-8"?><CAEXFile
FileName="ET200SP_with_ET200AL_ExtensionRack_And_SideBySideConnection.aml"
SchemaVersion="2.15" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
...
<InternalElement ID="aa0f752d-63fb-4f8d-b840-72d39f2ae508" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ET-Con</Value>
</Attribute> ...
<InternalElement ID="695f110e-2c2b-435f-b3c1-cdc77d83d559" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute> <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="87b9a1a3-dab8-4c62-8627-fbc8ab84e8f7"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement><SupportedRoleClass
RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
<InternalElement ID="124371c4-52ca-4ecc-a476-84e7bc1439b8" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ET-Con</Value>
</Attribute>
...
<InternalElement ID="0e354480-219c-4178-9f21-21fb369d446e" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X30</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="ab03743a-59a5-490e-b6c6-ce5359e345fd"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
...
</InternalElement>
<InternalElement ID="a0ff4884-cb81-40ed-85c8-823650687a8f" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X31</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>3</Value>
```

```
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="33caac7c-2944-4b1d-9fb8-8a451e4c1c25"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<InternalElement ID="d1aa8afe-e20a-4b82-996a-9fca151b2c51" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ET-Con</Value>
</Attribute>
...
<InternalElement ID="e713beda-1d1c-48d7-8674-8640080c87f7" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X30</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="046e2a6a-dca9-4a63-8f4f-53cabac1a65c"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="81999cc0-694f-4f7b-ba12-72ee9010cb11" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X31</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>3</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="e611d310-9ff7-45bb-b379-36e9cbea5080"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
```

```
...

<InternalElement ID="d21e9e3d-f5ea-4992-87b4-1598ecd2e159" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ET-Con</Value>
</Attribute>
...
<InternalElement ID="74cff4f7-3023-4235-9255-b36bc9b3e3e6" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X30</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="5322297a-a9c6-474d-82d0-992234730926"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalElement ID="c671263b-4c6e-4914-8a29-cec88bc917bd" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string"><
Value>ET-Con</Value>
</Attribute>
...
<InternalElement ID="b220aab7-ed61-481e-a9e5-5588b510c46c" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X30</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="ef830254-217a-42b3-b2f1-2363acf79b69"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<InternalLink Name="Link To Port_1" RefPartnerSideA="695f110e-2c2b-435f-b3c1-
cdc77d83d559:CommunicationPortInterface"
RefPartnerSideB="0e354480-219c-4178-9f21-21fb369d446e:CommunicationPortInterface" />
```

```
<InternalLink Name="Link To Port_2" RefPartnerSideA="a0ff4884-
cb81-40ed-85c8-823650687a8f:CommunicationPortInterface"
RefPartnerSideB="e713beda-1d1c-48d7-8674-8640080c87f7:CommunicationPortInterface" />
<InternalLink Name="Link To Port_3" RefPartnerSideA="81999cc0-694f-4f7b-
ba12-72ee9010cb11:CommunicationPortInterface" RefPartnerSideB="74cff4f7-3023-4235-9255-
b36bc9b3e3e6:CommunicationPortInterface" />
<InternalLink Name="Link To Port_4"
RefPartnerSideA="0201304e-39a9-4c9a-8316-2dd86fd95d0c:CommunicationPortInterface"
RefPartnerSideB="b220aab7-ed61-481e-a9e5-5588b510c46c:CommunicationPortInterface" />
...
```

Note

For side by side connections extension rack interface shall be exported only when modules were plugged next to each other. Also, the number of dummy ports added under extension ET-Con dummy interface is based on the ports participating in extension rack connection at module level. In above example, "DI_02" module is plugged in between modules "DI_01" and "DI_03" modules, so in exported AML file corresponding ET-Con dummy interface has two dummy ports. But incase of "AI_01" module which does not have any side by side modules. So, in this case "AI_01" module which is not having any ET-Con dummy interface details. Importing the AML file back to an empty TIA Portal project shall create the same device having multiple racks with extension rack connections.

- For ET200 AL and ET200SP modules, TIA Portal support default connection between primary rack and extension racks when relevant modules are plugged. In such case, connection information from AML file has no relevance during Import.
- Side by Side connection descriptions in the AML file has no relevance during the import since the side by side connections are automatically established when there are two adjacent modules imported into the same rack.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.36 Export/Import of PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open.
  See Opening a project (Page 128)
- PLC is offline.

Application

Exported and imported symbols and tags are assigned to a device item. CAx import/export concerns hardware oriented symbols and tags. The symbols and tags are exported only with the controller target device item, e. g. the CPU and not with other device items they might refer to, e. g. an I/O module. Like devices, the tags are often grouped in tag tables and in a hierarchical folder structure.

AML structure elements

PLC tags, tag tables and tag user folders can be exported and imported via CAx import/export function. The tag object are mapped in the following AML structure elements:

- `<InternalElement>`
  Tab tables and tag user folders are mapped as internal elements of the related PLC with the respective role class.
- `<ExternalInterface>`
  Represents a PLC tag, dedicated to the internal element of the related tag table or tag user folder.

A mapping channel with a PLC tag is exported as communication partner via the `<internal link>` element. The following XML structure shows an example:

```
...
    <InternalLink Name="Link To Tag_1"
       RefPartnerSideA="b33451f6-d88f-4900-8dbe-41f1be1e3535:Channel_DI_0"
       RefPartnerSideB="b2b937ee-d5db-4826-9340-027b1da22828:Tag_1" />
...
```

PLC tag user folder

The objects "TagUserFolder" only need the "Name" attribute in CAx import and export files.

Attributes of a PLC tag table

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory, ignored if "AssignToDefault" = TRUE	
AssignToDefault	Import-only	Used to identify the default tag table during import. If "AssignToDefault" = TRUE, all tags are created under the default tag table of the TIA Portal.

Attributes of a PLC tag

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	
DataType	Mandatory	
LogicalAddress	Optional	Imported and exported in international mnemonics format
IoType	Optional	Input or output
Comment	Optional	

Note

TIA Portal V16 supports export and import of IoType attribute in AML file with AR APC V1.1.0

Example: AML structure

The following figure shows the structure of the following exported tag objects:

- empty default tag table
- tag user folder "Group_1"
- included tag table "Tag table_1
- four tags

```
<InternalElement ID="a310b193-ba04-49d7-a8e3-004619c7d9d2" Name="Default tag table">
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable" />
</InternalElement>
<InternalElement ID="0feff703-9c70-4ca9-b3b3-8de8229696dd" Name="Group_1">
    <InternalElement ID="f92g9ce4-c015-459f-9f59-8f94bca3b186" Name="Tag table_1">
      <ExternalInterface ID="fc0c8c5a-fd5b-443b-b430-6435b6aa22ff" Name="Tag_1" RefBaseClassPath="AutomationProjectConfigurati
        ...
      </ExternalInterface>
      <ExternalInterface ID="450d6a1d-81b8-49ae-a104-c0072933d669" Name="Tag_2" RefBaseClassPath="AutomationProjectConfigurati
        ...
      </ExternalInterface>
      <ExternalInterface ID="3de17a36-b5c5-4fc7-9fc3-47e4a8f95087" Name="Tag_3" RefBaseClassPath="AutomationProjectConfigurati
        <Attribute Name="DataType" AttributeDataType="xs:string">
          <Value>Word</Value>
        </Attribute>
        <Attribute Name="IoType" AttributeDataType="xs:string">
          <Value>Input</Value>
        </Attribute>
        <Attribute Name="LogicalAddress" AttributeDataType="xs:string">
          <Value>IW0</Value>
        </Attribute>
      </ExternalInterface>
      <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
</InternalElement>
...
```

See also

Structure of the CAx data for importing/exporting (Page 1435)

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.37 Export/Import of RH/PLC

Requirement

- The TIA Portal Openness is connected to theTIA Portal
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- PLC is offline

Application

You can use TIA Portal to export and import AML file following AR APC V1.1 with R/H PLC having same IP Address configuration.

Attribute

TIA Portal supports only one attribute in AML file for R/H PLC, if available in TIA Portal user interface:

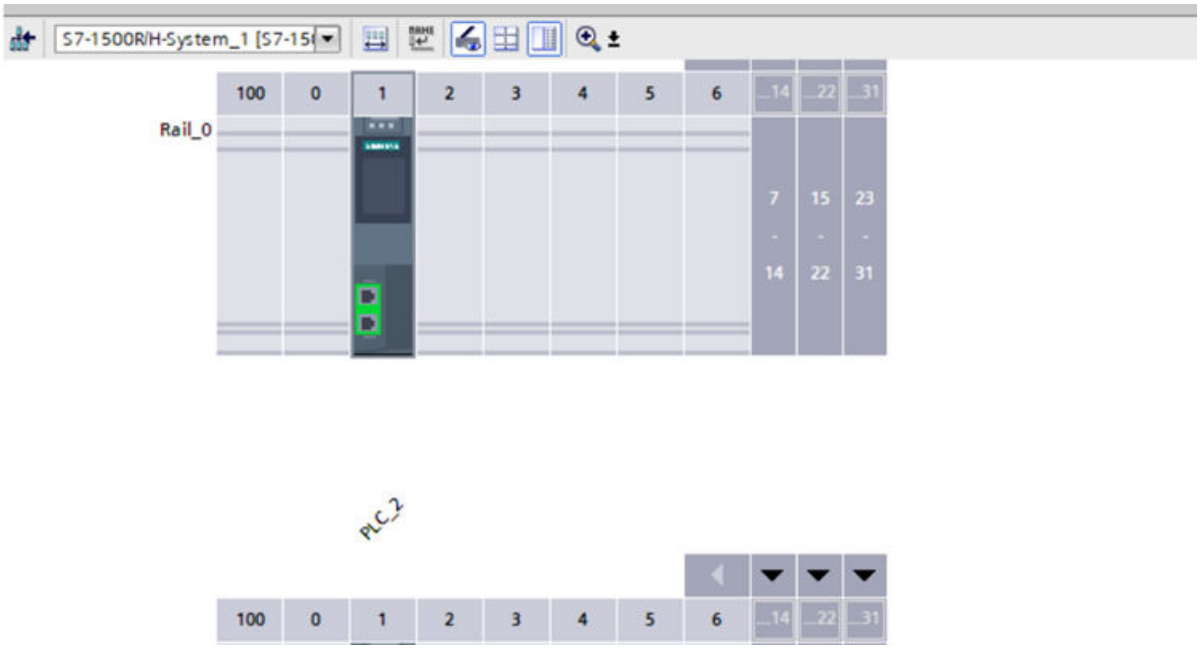
Device Item Attribute Name	Handling	Comment
HNetworkAddress	Mandatory	Exported/Imported, if the device item R/H PLC supports this attribute in TIA Portal and is non empty otherwise it shall be skipped. For import, the "Enable the system IP address for switched connection" shall be set and the HNetworkAddress shall be valued.

Restriction

- Export
  - Only if the "Enable the system IP address for switched connection" in TIA Portal is selected.

Example: Exported configuration

The following configuration shows a device item where the HNetworkAddress are configured.



AML structure of export file

The following figure shows the structure of exported AML file for R/H PLC

```
<InternalElement ID="e1879cf8-8222-4ce3-b171-60c56aed7f18" Name="PROFINET interface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32768</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="a32c67c7-47dc-4362-a9c4-b41b93b58eff" Name="E1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<Attribute Name="HNetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.3</Value>
</Attribute>
<ExternalInterface ID="802676fa-212f-4bf5-b112-de94993a0340" Name="LogicalEndPoint_Node"
RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.38 Export/Import AML with customized tag and deviceitems

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- PLC is offline

Application

You can use TIA Portal to export and import AML file following AR APC V1.1 with 'Customized' sub-attribute of tag and deviceitem.

Attribute

The following table shows the related attribute available for a tag during CAx import and export files:

Attribute Name	Handling	Comment
Customized	Optional for export/ import	Sub-attribute of 'DataType' of a tag. True and False are the only allowed values for Customized. While importing, the attribute customized has no relevance During export Customized sub-attribute has a value "true" for datatypes other than IEC 61131 datatypes. For IEC 61131 compliant datatypes the Customized sub-attribute shall not be exported.

The following table shows the related attribute available for a deviceitem during CAx import and export files:

Attribute Name	Handling	Comment
Customized	Optional for import	It is a child attribute to parent 'DeviceItemType' attribute for a DeviceItem. True and False are the only allowed values for Customized. While importing and exporting, Customized attribute has no relevance

Exported AML file

The below AML fiile shall be generated during export of an AML file with customized deviceItems.

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile FileName="Project26.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
...
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="03ecf798-3e07-4976-b281-f8b98eb3a590" Name="Project26">
...
<InternalElement ID="c218aea9-93b8-4719-be6f-ccfa9517e2c6" Name="S71500/ET200MP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.S71500</Value>
</Attribute>
<InternalElement ID="c1ae306f-183f-47b8-91e6-cb331c559278" Name="Rail_0">
...
<InternalElement ID="8d7d5ee1-603a-4897-b47e-8954d4c21a31" Name="PLC_1">
...
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>CPU</Value>
</Attribute>
...
<InternalElement ID="a2a7f0bc-068c-4904-84b1-73ed87e28de1" Name="Tag table_1">
...
<Attribute Name="DataType" AttributeDataType="xs:string">
<Value>Conn_Any</Value>
<Attribute Name="Customized" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
</Attribute>
</InternalElement>
...
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
AutomationProject" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.39 Export/Import of FailSafe PLC

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- PLC is offline

Application

You can use TIA Portal to export and import an AML file following AR APC V1.1 with FailSafe PLC having failsafe attributes.

Attributes

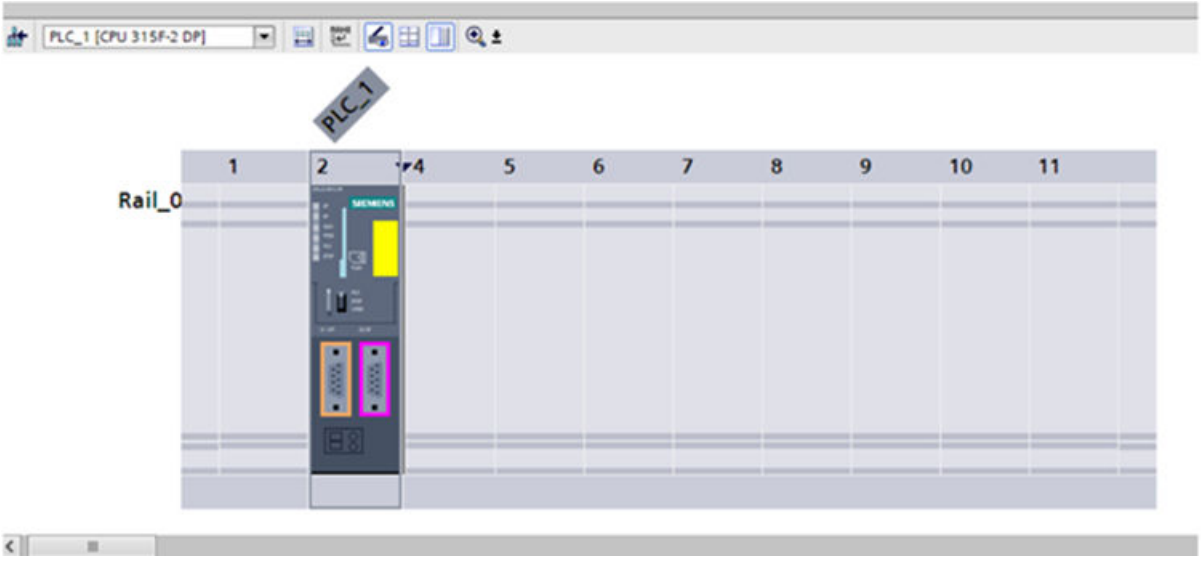
The following table shows the list of failsafe attributes for CAx import and export AML files:

Openness Attribute	Handling	Comment	AR APC Name in AML
Failsafe_FSourceAddress	Mandatory	Export/import only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_FSourceAddress
Failsafe_LowerBoundForFDestinationAddresses	Mandatory	Exported/Imported only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_LowerBoundForFDestinationAddresses
Failsafe_UpperBoundForFDestinationAddresses	Mandatory	Exported/Imported only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_UpperBoundForFDestinationAddresses

Openness Attribute	Handling	Comment	AR APC Name in AML
Failsafe_CentralFSourceAddress	Optional	Exported/Imported only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_FSourceAddress
Failsafe_FDestinationAddress	Optional	Exported/Imported only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_FDestinationAddress

Example: Exported configuration

The following configuration shows a device item where the attributes are configured.



AML structure of the export file

The following figures show the structure of the exported AML file.

```
<InternalElement ID="9c944d2c-e0ae-4f39-b35a-a63faaf35be7" Name="PLC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>CPU 1511TF-1 PN</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>CPU</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 511-1UK01-0AB0</Value>
</Attribute>
<Attribute Name="InstallationDate" AttributeDataType="xs:dateTime">
<Value>2019-02-28T08:12:12.987Z</Value>
</Attribute>
<Attribute Name="Failsafe_FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_LowerBoundForFDestinationAddresses"
AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_UpperBoundForFDestinationAddresses"
AttributeDataType="xs:string">
<Value>99</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V2.8</Value>
</Attribute>
<InternalElement ID="4f718e93-b541-4983-8f13-1f5b21c3e70c" Name="Default tag table">
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
TagTable"/>
</InternalElement>
<InternalElement ID="20e19f5b-8ace-4e0c-af0b-c710ae4817da" Name="CPU display_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>3</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="5a24516f-17d6-4b2a-a4ac-efc1b577875d" Name="Card reader/writer_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>4</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
```

```
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement> <InternalElement ID="0f746d71-035e-4e64-b0d7-51d0449cfd88" Name="OPC
UA_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>254</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value> </Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="a0633104-a2ac-4680-bb99-81df50f5ec40" Name="PROFINET interface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32768</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="e3497176-dbba-4fec-9d3a-772ae13987c4" Name="E1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value> </Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<InternalElement ID="3208384f-d5ba-4ccb-b8da-f08ec38ec681" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32769</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="4a47c05e-9656-4e02-9b51-23b065b6fe47" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32770</Value>
</Attribute>
```

```
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="e3fdb611-4b68-4682-b154-ae43c74a24d3" Name="F-DI 16x24V DC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>F-DI 16x24V DC</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value> </Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 526-1BH00-0AB0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V1.0</Value>
</Attribute>
<InternalElement ID="77c4fea0-baba-44e6-80f2-72b7b830a88a" Name="F-DI 16x24V DC_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute> <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<Attribute Name="Failsafe_FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_FDestinationAddress" AttributeDataType="xs:string">
<Value>655</Value>
</Attribute>
</InternalElement>
</InternalElement>
```

Extended role for device item

AR APC 1.2 recommends following changes for CAx exchange in TIA Portal V17 onwards, any device item shall be exchanged with one additional role.

'AutomationProjectConfigurationProfiSafeRoleClassLib/DeviceItemProfiSafe' along with the existing role 'AutomationProjectConfigurationRoleClassLib/DeviceItem' in case AR APC recommended failsafe attributes are supported.

The device item shall be exported with additional role in case it supports extended attributes like Failsafe_FSourceAddress, Failsafe_LowerBoundForFDestinationAddresses, Failsafe_UpperBoundForFDestinationAddresses, Failsafe_CentralFSourceAddress,

Failsafe_FDestinationAddress. And it shall be possible to import device item with or without extended role also.

AML file with device item

Below XML snippet depicts the AML file with "device item" supporting failsafe attributes module with an extended role.

```
<InternalElement ID="9d6c270a-2a48-426a-9dae-8cf88c5a591a" Name="PLC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>CPU 414F-3 PN/DP</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>CPU</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 414-3FM06-0AB0</Value>
</Attribute>
<Attribute Name="Failsafe_FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_LowerBoundForFDestinationAddresses"
AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_UpperBoundForFDestinationAddresses"
AttributeDataType="xs:string">
<Value>99</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V6.0</Value>
</Attribute>
...
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationProfiSafeRoleClassLib/
DeviceItemProfiSafe" />
</InternalElement>
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.40 Export/Import of Failsafe IO

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a project (Page 128)
- PLC is offline

Application

You can use TIA Portal to export and import an AML file following AR APC V1.1 with FailSafe IO having failsafe attributes.

Attributes

The following table shows the related attributes available on Failsafe IO module/submodule for CAx import and export files :

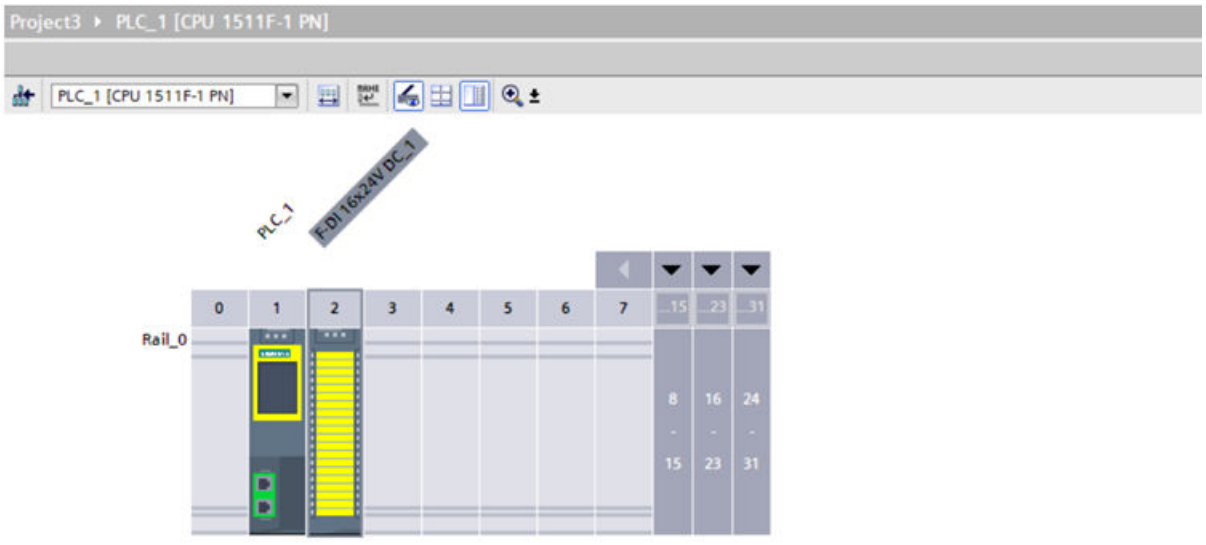
Device Item Attribute Name	Handling	Comment
FSourceAddress	Mandatory	Export/import only if the device item is a failsafe IO and supports in TIA Portal.
FDestinationAddresses	Mandatory	Export/Import only if the device item is a failsafe IO and supports in TIA Portal.

Note

- For most of the Failsafe IO's, FSourceAddress is a ReadOnly it is not writable and is a reflection of FCentralFSourceAddress attribute of failsafe PLC. The same value shall get exported and ignored during import.
- If the FSourceAddress and FDestinationAddress are Readonly, then value shall be exported. However, during import warning message should be displayed in info tab of TIA Portal.
- FSourceAddress and FDestinationAddress should be supported for IO module and sub module level.

Example: Exported configuration

The following configuration shows a device item where the attributes are configured.



AML structure of the export file

The following figure shows the structure of exported AML file.

```
<Attribute Name="FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="FDestinationAddress" AttributeDataType="xs:string">
<Value>65534</Value>
</Attribute>
```

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.41 Export/Import vendor specific attribute

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
  See Connecting to the TIA Portal (Page 82)
- A project is open
  See Opening a Project (Page 128)

Application

In TIA Portal, you can export and import an AML file for device and deviceitems having "Manufacturer" attribute so that you can exchange vendor specific information in round trip scenarios.

The attribute "Manufacturer" is not supported by Openness. It shall be consumed through "Transformation Plugin" during Export/Import. AML file with AR Drive recommendation document of version 1.0 is supported.

Note

AML files shall be generated(exported) by TIA Portal V16 following AR APC V1.1.

Export

You can export the "Manufacturer" attribute into AML file, if it is available in AmiModel, which shall be filled by "Transformation Plugin" users.

Import

You can import "Manufacturer" attribute to TIA Portal, wherein vendor specific attribute is available in AmiModel which shall be consumed by "Transformation Plugin" users.

Currently this attribute shall be supported only for 'Drive' devices and device items since only they do have official "Transformation PlugIn" written for CAx.

See also

Connecting to the TIA Portal (Page 82)

Opening a project (Page 128)

6.5.42 AML attributes versus TIA Portal Openness attributes

Access attributes and export/import attributes

Via TIA Portal Openness you can access attributes of hardware objects. Single names you use to access these attributes e. g. of a device item differs from the attributes names in the export/import AML file.

Attributes list

The following table provides an overview to both kinds of attributes:

Table 6-6 Attribute names of devices and GSD/GSDML devices

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
Comment	Comment

Table 6-7 Attribute names of device items

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	Mapped to substring of <TypeIdentifier> (i.e., value before first "/" operator) ignoring firmware version part in it. Mapping substring is applicable only when TypeIdentifier starts with <OrderNumber:> prefix and it has firmware version part otherwise mapped to complete <TypeIdentifier>.
FirmwareVersion	<FirmwareVersion> mapped to substring of <TypeIdentifier> (i.e., value after first "/" operator). Mapping substring is applicable only when <TypeIdentifier> starts with <OrderNumber:> prefix and it has firmware version part.
TypeName	TypeName
DeviceItemType (for CPU and HeadModule)	**Classification**
PositionNumber	PositionNumber
BuiltIn	**IsBuiltIn**
PlantDesignation IEC	**PlantDesignation**
LocationIndentifier IEC	**LocationIdentifier**
Comment	Comment

Table 6-8 Attribute names of GSD/GSDML device items

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
TypeName	TypeName
DeviceItemType (for HeadModule)	**Classification**
PositionNumber	PositionNumber
BuiltIn	**IsBuiltIn**
Comment	Comment
Label	Label

Table 6-9 Attribute names of tags

AML file	TIA Portal Openness
Name	Name
DataType	**DataTypeName**
LogicalAddress	LogicalAddress
Comment	Comment

Table 6-10 Attribute names of tag tables

AML file	TIA Portal Openness
Name	Name
AssignToDefault	**IsDefault**

Table 6-11 Attribute names of addresses

AML file	TIA Portal Openness
StartAddress	StartAddress
Length	Length
IoType	IoType

Table 6-12 Attribute names of ports

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
PositionNumber	PositionNumber
BuiltIn	**IsBuiltIn**

AML file	TIA Portal Openness
Comment	Comment
Label	Label

Table 6-13 Attribute names of devices with IO-interface

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
DeviceItemType (for CPU and HeadModule)	**Classification**
PositionNumber	PositionNumber
BuiltIn	**IsBuiltIn**
Label	Label
Comment	Comment

Table 6-14 Attribute names of channels

AML file	TIA Portal Openness
Type	Type
IoType	IoType
Number	Not mapped to any attribute in TIA Portal Openness.
Length	**ChannelWidth**

Major Changes

7

7.1 Major changes for long-term stability in TIA Portal Openness V19

Changes

If you have considered the hints concerning programming across versions and do not rebuild your Openness application to V19, your application will run without any restrictions on any computer even if only a TIA Portal V19 is installed.

If you rebuild your Openness application to V19, it is necessary to recompile your application using the Siemens.Engineering.dll of V19. In some cases, it might be necessary to adapt the code of your application.

CAx/AML Data exchange

Changes in behaviour of command line facility

With TIA Portal V19 onwards, the command line facility for CAx export and import is no longer available to the user.

Change in behaviour of Profinet / Ethernet port handling

With TIA Portal V19 onwards, Label value for Profinet / Ethernet interface ports would be exported without space irrespective its value in TIA Portal. For example: If Label = P1 R, AML file would have Label = P1R

Change in behavior of IO Link port handling

With the TIA Portal V18 Update 2 onwards, IO Link port shall be exchanged with Label value as 'C/Q<n>' where n is the port number. for example: C/Q1, C/Q2 etc.,
For AML exchange of IO Link configuration in TIA Portal V18 Update 2 (via S7-PCT), use "S7-PCT 3.5 SP3 Update 3" version or higher.

WebServerUserManagement

Changes in behaviour of removal/disable of existing Openness services

With the TIA Portal V19, the Centralized UMAC functionality will be active, but the WebServerUserManagement will be disabled for all PLC firmware versions from V3.1 onwards.

However, until TIA Portal V18, the WebServerUserManagement will be operational, while the Centralized UMAC functionality will be inactive for PLC firmware versions up to V3.0.

UMAC on PLC

Changes in behaviour of online legitimation call via UMAC on PLC

If the Openness application is running with a TIA Portal Openness API <= 18, the online legitimation via Openness is still possible via the download and upload configurations. But only if the used PLC is protected by a legacy ProtectionLevel legitimation.

With TIA Portal Openness API >= V19, all legitimation calls, especially the new UMAC on PLC, will be delivered to the event handler of the online legitimation event of the ConnectionConfiguration class. In case the OnlineAuthenticationConfiguration type is not handled there by the user code, there will be a second call to the callback method of the particular feature (e.g., download or upload). But there only the old protection mechanisms (until V18) can be handled.

Enforcement of strict password policies

Changes in behaviour of strict password policies for know how protection of blocks

With TIA Portal V19, the Strict password policies are applied while setting a password. This also applies to TIA Portal Openness calls, especially setting know-how protection. An exception will be thrown if the password does not follow the policy.

WinCC Unified Screen Editor

Changes in behaviour of MultilingualText items in context to WinCC Unified Screen Editor

All MultilingualText items are changed from unformatted to formatted in scope of the WinCC Unified Screen Editor since TIA Portal V18.
This means all texts must be set formatted from now on. Plain text will be rejected by throwing an exception. for example:

```
Language language = project.LanguageSettings.Languages.Find(new CultureInfo("en-US"));
MultilingualText multilingualToolTipText1 = ((HmiButton)screenItem1).ToolTipText;
MultilingualTextItem multilingualTextItem1 = multilingualToolTipText1.Items.Find(language);
multilingualTextItem1.Text = "<body><p>Modified button text from Openness</p></body>";
```

Support of Blocks/UDTs

Changes in schema to support NamedValueTypes usage in Blocks

In TIA Portal V19, the support for Named Value Types has been introduced (only for Software Units in S7-1500 PLCs).
In Openness, a new scope "NamedValueConstant" is introduced in the SimaticML schema from V19 to support the usage of a Named Value Type in Program Blocks or PLC Data Types. The Named Value Types which are used in PLC programming artifacts (Program Blocks, PLC

Data Types) are visible in the new scope "NamedValueConstant" during import/export via SimaticML.

```
<Access Scope="NamedValueConstant" UId="27">
<Constant Name="_.siemens.simatic.Named_value_type_1#UNDEFs" UId="28"/>
</Access>
<Token Text=";" UId="31"/>
<NewLine Num="2" UId="32"/>
```

7.2 Major changes for long-term stability in TIA Portal Openness V17

Changes

If you have considered the hints concerning programming across versions and you do not rebuild your Openness application to V17 your applications will run without any restrictions on any computer even if only a TIA Portal V17 is installed.

If you rebuild your Openness application to V17 it is necessary to recompile your application using the SiemensEngineering.dll of V17. In some cases it might be necessary to adapt the code of your application.

Change in implementation of GetAttribute

Up to TIA Portal Openness V16, the visibility of the implementation of GetAttribute/SetAttribute/GetAttributeInfos is dependent on the fact whether the type declares as "Supports dynamic attribute". To access methods like GetAttribute/SetAttribute/GetAttributeInfos, Then the type should support dynamic attributes. When the type does not support dynamic attributes, then you have to explicitly cast to IEngineeringObject to get the methods GetAttribute/SetAttribute/GetAttributeInfos.

The below code sample shows how the code is executed up to TIA Portal Openness V16:

```
var projectAttributeInfos = ((IEngineeringObject) project).GetAttributeInfos();
```

With TIA Portal Openness V17, all methods of IEngineeringObject is implicitly implemented on all types that implement IEngineeringObject so that the explicit cast is not required, all methods will be visible to you. The change will be available in all previous version engineering dll's i.e V15, V15.1 & V16 supported in V17.

For following methods, typecast is not required:

- **object GetAttribute(string name) -** Gets an attribute with the given name.
- **IList<EngineeringAttributeInfo> GetAttributeInfos()** - Returns a collection of EngineeringAttributeInfo objects describing the different attributes on this object.
- **IList<object> GetAttributes(IEnumerable<string> names)** - Gets a list of attributes for the given names.

- **void SetAttribute(string name, object value)** - Sets an attribute with the given name to the given value.
- **void SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes)** - Sets the attributes with the given names to the given values as indicated in attribute

By introducing this change, there will be a slight change-in behavior if you recompile the code used under extension method. For example when you have used extension method and when the code is compiled into the V17 engineering dll, call from the extension method doesn't get executed. In this case you would have to change code to stay with the behavior compatible. There would also be no compiler error.

The below example code explains where GetAttribute method doesn't get executed:

```
static class CustomerExtension
{
public static object GetAttribute(this Project project, string name)
{
// Customer Logic
return ((IEngineeringObject)project);
}
}
```

Changes on AssignInterface API

Before TIA Portal Openness V17, the AssignInterface API cannot be not used inside a transaction in TIA Portal Openness application. With TIA Portal Openness V17, this behaviour is changed and AssignInterface API can be used in transaction.

Setting master system number

With TIA Portal Openness V17, it is possible to set the master system number during the import of a API.

Removal of ExternalSourceCreate method

Before TIA Portal Openness V17, the Create action from ExternalSource is supported with "Not Supported Exception" in TIA Portal Openness application. With TIA Portal Openness V17, the Create action from external sources composition will be removed.

Removal of attribute SyncRole

Before TIA Portal Openness V17, the attribute "SyncRole" is read-only for IPC627D device. With TIA Portal Openness V17, the attribute "SyncRole" will be removed from device where the attribute serves no purpose.

Change in behaviour of Safety Compile

Before TIA Portal Openness V17, Failsafe with a set password and without a user being logged in is not supported and throws compile error result. With TIA Portal Openness V17, this behaviour is changed from compile error to an exception.

Removal of download configuration

With TIA Portal Openness V17, STEP7 specific download configurations are no longer available in TIA Portal Openness without STEP7 installed.

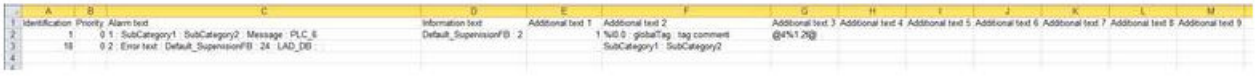
Restriction concerning attribute "SecondaryType" when importing OB MC-Transformation

With TIA Portal Openness V17, when importing the OB MC-Transformation, the attribute "SecondaryType" needs to have the exact string value "Transformation" (which is also the value that is exported).

Export ProDiag Information

With TIA Portal Openness V17, you can export ProDiag alarm messages within a ProDiag FB via public Openness API in CSV format.

The output can be viewed in Microsoft Excel.



Changes in behavior of ExportInstanceTextsToXlsx and ExportToXlsx method in PlcAlarmTextProvider

Before TIA Portal Openness V17, if the file (given as path parameter) already exists, the file will be overwritten during the export. With TIA Portal Openness V17, this behaviour is changed and if the file (given as path parameter) already exists, UserException will be thrown to the user.

Changes in behavior of Export API to accept specific path

Before TIA Portal Openness V17, In Export API, specific paths are not allowed. EngineeringTargetInvocationDetailException.SpecificPathException exception will be thrown when specific path is specified.

With TIA Portal Openness V17, this behavior is changed and the specific paths are allowed.

Below are the scenarios where TIA Portal Openness API allows specific path.

Input path	TIA Poral Openness V17
D:SampleDirectory\Sample	No exception is thrown
D:\SampleDirectory/Sample	No exception is thrown
D:\SampleDirectory\Sample	No exception is thrown
D:\\SampleDirectory\Sample	No exception is thrown
D:\\\SampleDirectory\Sample	No exception is thrown
D:\\SampleDirectory\\Sample	No exception is thrown
D:\\\\\\SampleDirectory\\\Sample	No exception is thrown
D:\\SampleDirectory\\\Sample	No exception is thrown
\\SampleDirectory\Sample	No exception is thrown
\\SampleDirectory\\Sample	No exception is thrown

Input path	TIA Poral Openness V17
\\\\SampleDirectory\\\\Sample	No exception is thrown
\\\\SampleDirectory\\\\Sample	No exception is thrown
//SampleDirectory/Sample	No exception is thrown
"SampleDirectory"	EngineeringTargetInvocationDetailException.SpecificPathException is thrown from export Openness API.
"//SampleDirectory"	EngineeringTargetInvocationDetailException.SpecificPathException is thrown from export Openness API.
"D:\SampleDirectory\Sample\..\otherSample"	EngineeringTargetInvocationDetailException.SpecificPathException is thrown from export Openness API.
"SampleDirectory\Sample"	EngineeringTargetInvocationDetailException.SpecificPathException is thrown from export Openness API.
"..\Sample", "SampleName"	EngineeringTargetInvocationDetailException.SpecificPathException is thrown from export Openness API.
"D:\ SampleDirectory\Sample"	EngineeringTargetInvocationDetailException.SpecificPathException is thrown from export Openness API.
D:\SampleDirectory\ \Sample	EngineeringTargetInvocationDetailException.SpecificPathException is thrown from export Openness API.
D:\SampleDirectory\Sample	EngineeringTargetInvocationDetailException.SpecificPathException is thrown from export Openness API.

Changes in data type of ScreenNumber attribute

With TIA Portal Openness V17 Siemens.Engineering.dll, the data type of ScreenNumber attribute is changed from Byte to UInt16.

Changes in behavior of CAx Export/Import

With TIA Portal Openness V17, following changes are done for CAx export/import:

- Support for custom(user specific) attributes on GSD/GSDML devices.
- An user can exchange CAx data on multiuser projects and UMAC protected projects.
- Support for exchange of CAx data with normalized TypeIdentifier. This facility is controlled via CAx user interface
- The structure of PN PN Coupler device inside AML file has changed. This is to avoid the internal structural differences in the exchanging tools (TIAP and ECAD systems) and arrive at a common structure for exchange.
- Export/Import of Plus PC Stations are supported. Remark: Exchange with EPLAN and other Tools will not work, since the additional defined attributes are not defined in AR APC.
- During Import, user notification for start address/network address re-assignment in case it is adjusted due to internal operations of TIAP. For ex: Any subnet/io system connection failures, Address overlapping etc., . This facility is controlled via CAx user interface Settings.
- A warning user notification incase user imports an AML file with OMS and security enabled PLC's.
- A new attribute 'ProfinetDeviceName' for all Ethernet nodes is introduced.

- The device with pluggable profinet/profibus interface deviceitem is always exported with 'segregated' roles. A pluggable interface deviceitem in TIA project shall be exported as a pluggable device item with DeviceItem role (having a 'new' child device item with CommunicationInterface role). However, Import of older AML files shall be supported always.
- Supports more tolerant (independent of IoType order within deviceitem) start address import.
- Import of compact module using 'TemplateReference' with some restrictions supported.

Support of blocks of new programming language CEM

With TIA Portal Openness V17, the new programming language CEM (Cause Effect Matrix) was introduced. If you encounter an instance of such a block accessing the attribute Programming Language in any Siemens.Engineering.dll < V17 will throw an exception. In Engineering.dll V17, the correct programming language is returned. Blocks of programming language CEM do not support the Export-method in any Siemens.Engineering.dll.

Support of new library types "Unified Faceplate" and "Unified Graphic"

With TIA Portal Openness V17, the new library types "Unified Faceplate" and "Unified Graphic" was introduced. These types are not supported in Openness. If you encounter instances of these types all versions of Siemens.Engineering.dll will thrown an exception.

7.3 Major changes in TIA Portal Openness V16

Changes

If you have considered the hints concerning programming across versions and you do not upgrade your project to V16 your applications will run without any restrictions on any computer even if only a TIA Portal V16 is installed.

If you upgrade your project to V16 it is necessary to recompile your application using the SiemensEngineering.dll of V16. In some cases it might be necessary to adapt the code of your application

Export of datablocks with export option "None"

As of TIA Portal Openness V16 members of a datablock which are read only will be exported as informative items. It is not longer possible to change these attributes in the exported xml.

I&M and Profisafe address attributes of pushbutton panel and key panel

As of TIA Portal Openness V16 I&M and Profisafe address attributes of pushbutton panel and key panel will be accessible at module level.

Export DB attributes of memory layout property

As of TIA Portal Openness V16 IDB of FBs, ArrayDBs and Graph blocks attributes of memory layout property will be exported with ReadOnly="True".

Access attribute in units

As of TIA Portal Openness V16 inconsistent import and export of "Access" attribute is supported in blocks, UDTs and tag table within units

SimaticML XML file's schema definition

As of TIA Portal Openness V16 boolean attribute node's SystemDefined attribute's default value become false in SimaticML XML file's schema definition. However, the change doesn't affect any XML export/ import feature. It is only important for users who try to generate XMLs using the schema file.

Enhance XML import method of blocks and types

As of TIA Portal Openness V16 extend the new import method with additional parameters (i.e. path, importOptions, and swImportOptions) and a new enum value IgnoreUnitAttribute. By using swImportOptions.IgnoreUnitAttributes, you can import block from unit to non-unit composition.

AML GUIDs are stored in App IDs

As of TIA Portal Openness V16 AML GUIDs are stored in CustomIdentity (App IDs) attribute instead of Comment to support round trip exchange of device and module.

Name of HMI Object classes

As of TIA Portal Openness V16, the following table shows the list of required name changes for HMI Object classes:

Class Name	New Class Name
AnalogAlarmComposition	HmiAnalogAlarmComposition
DiscreteAlarmComposition	HmiDiscreteAlarmComposition
AlarmClassComposition	HmiAlarmClassComposition
DataLogComposition	HmiDataLogComposition
AlarmLogComposition	HmiAlarmLogComposition
LoggingTagComposition	HmiLoggingTagComposition
LogConfiguration	DataLog

Addition/Removal of properties name for DataLog/Alarm Log

As of TIA Portal Openness V16, the following new properties "StorageDevice" and "StorageFolder" are added to DataLog/AlarmLog and properties such as "StoragePath" and "RequireExplicitRelease" are removed from DataLog/AlarmLog and ScreenItems respectively.

You can find the updated code example for StorageDevice and StorageFolder properties as below:

```
HmiDataLog dataLog = hmiSoftware.DataLogs.Find("DataLog1");
dataLog.Settings.StorageDevice = DeviceNode.Local;
dataLog.Settings.StorageFolder = @"D:\workdir\DataLogs";
```

Remove a property name from HMI Tag

As of TIA Portal Openness V16, the 'DisplayName' property is removed from the HMI Tag object.

7.4 Major changes in TIA Portal Openness V15.1

Changes

If you have considered the hints concerning programming across versions and you do not upgrade your project to V15.1 your applications will run without any restrictions on any computer even if only a TIA Portal V15.1 is installed.

If you upgrade your project to V15.1 it is necessary to recompile your application using the SiemensEngineering.dll of V15.1. In some cases it might be necessary to adapt the code of your application

Type identifiers

The type identifier for racks and devices of "PC with ports" and "Ethernet devices with ports" have been renamed.

PC with ports	Type identifier before V15.1	Type identifier as of V15.1
Device	System:DesktopPC.Device	System:Device.DesktopPC
Rack	System:DesktopPC.Rack	System:Rack.DesktopPC
Device item	System:DesktopPC.Port<X> <X> denotes the number of ports	System:DeviceItem.EthernetDe-vice.Port<X> <X> denotes the number of ports

Ethernet devices with ports		
Device	System:DummyPC.Device	System:Device.EthernetDevice
Rack	System:Rack.DummyPC	System:Rack.EthernetDevice
Device item	System:DummyPC.Port<X> <X> denotes the number of ports	

Failures when trying to connect to TIA Portal

The messages in case of failures when trying to connect to TIA Portal have been enhanced in a more specific way.

Cross-thread operations

Access to Openness objects is not inherently thread-safe.
If you use multi-threading to improve the performance of your Openness application, it is recommended to create your TIA Portal instance with an MTA.
If TIA Portal is created or attached within an STA thread, all Openness objects associated with that Portal instance must be accessed from the same STA thread; otherwise, an exception will be thrown.

Submodules do not have attributes Author and TypeName

The attributes Author and TypeName have been removed from submodules which cannot be plugged.

Opening of a global library

As of TIA Portal Openness V15.1 a global library can be opened via Openness independant of a persisted preview mode of the library.

Application exit codes

In case of an application exit code

- Up to TIA Portal Openness V15 you get a non recoverable exception
- As of TIA Portal Openness V15.1 you get a EngineeringRuntimeException or a EngineeringTargetInvocationException if the error code is known and a non recoverable exception if the error code is unknown.

Schema extension for nameless parameters

The import of SCL blocks is possible even if ENO is used at a nonformal call..

Header of indexed parameters

As of TIA Portal Openness V15.1 the header of indexed parameters my not be changed via Openness.

Some drive parameters are modelled as indexed parameters, as they provide several pieces of data for one topic. The modeling of the indexed drive parameters in Openness follows the drive specific definition of the respective parameters as they are defined in the respective list manual.

Indexed parameters are modeled as follows:
Header item: The respective drive parameter without any index. The header item contains a descriptive text, to inform you about the semantics of the referenced indexed drive parameter. Therefore the header item is readonly, because it holds no actual value.
Indexed Items: The indexed items of the drive parameter below the header item. They provide descriptive text, defining the semantics of the specific indexed item (readonly). In addition, they also provide the value that can be retrieved via Openness API. If the respective indexed parameter is also writeable, than the value can also be set via Openness API.

Attribute TransmissionRateAndDuplex

Some faulty enum values for attribute TransmissionRateAndDuplex have been corrected, for example the enum value "POFPCF100MbpsFullDuplexLD" has been removed and "POFPCF100MbpsFullDuplex" has been added. For detail information, please see Configurable attributes of a port-to-port connection (Page 196)

Attribute AutoNumber for know how protected blocks

As of V15.1 the attribute AutoNumber can not be changed via TIA Portal Openness if a block is know how protected.

Number of channels listed by ChannelInfo interfaace

Up to TIA Portal Openness V15 the ChannelInfo interface the number of available channels hasn't been correct for some mdoules.

Access to ProDiag FB attributes

The following attributes of a ProDiag FB can be accessauthored via TIA Portal Openness:

- Version
- Initial values acquisition
- Use central timestamp

Import/Export of failsafe blocks

Import of failsafe blocks from previous versions is not possible.

Export of system generated failsafe blocks will be prevented as of TIA Portal Openness V15.1.

R/H systems

Download or access for R/H devices is available at device.

Online and Download Provider are not available for individual R/H PLC (DeviceItems).

For PLC2 of an R/H system, SoftwareContainer will not be available.

7.5 Major changes in TIA Portal Openness V15

Changes

If you have considered the hints concerning programming across versions and you do not upgrade your project to V15 your applications will run without any restrictions on any computer even if only a TIA Portal V15 is installed.

If you upgrade your project to V15 it is necessary to recompile your application using the SiemensEngineering.dll of V15.

In some cases it is necessary to adapt the code of your application

- Behaviour changes for compositions in DeviceItemComposition
- BitOffset of ASi addresses
- Exception class
- System folders of system UDTs
- Submodules do not have attributes Author and TypeName
- Timestamp for last modification
- Export XML for GRAPH blocks
- Importing tag tables
- Modifying not failsafe relevant attributes of a PLC
- Modifying F-parameters while safety password is set
- Accessing TO objects in a S7 1200 CPU

Behaviour changes for compositions in DeviceItemComposition

The following compositions in DeviceItemCompositon have been changed for a dynamic behaviour. The composition is updated now if an element is added or deleted via the user interface of TIA Portal.

- IoSystem - ConnectedIoDevices
- Subnet - IoSystems
- Subnet - Nodes
- NetworkInterface - Nodes
- NetworkInterface - Ports
- NetworkPort - ConnectedPorts
- SubnetOwner - Subnets

BitOffset of ASi addresses

If a module has an input and an output address for both address objects the correct attribute BitOffset will be provided.

If a module has channels the attribute BitOffset will not be provided for the channel.

Exception class

ServiceID and MessageID have been removed from exception class

Submodules do not have attributes Author and TypeName

The attributes Author and TypeName have been removed from submodules which cannot be plugged.

System folders of system UDTs

For system folders of system UDTs the appropriate folder and composition is provided. This leads also to a change in the hierarchy of compare results.

Timestamp for last modification

If during an upgrade an object is changed the timestamp for the last modification is changed as well.

Export XML for GRAPH blocks

The export XML for GRAPH blocks contains an additional empty action: <Actions />

Importing tag tables

Setting tag attributes is not longer dependent from data types.

Modifying not failsafe relevant attributes of a PLC

All not failsafe relevant attributes of a PLC can be modified via TIA Portal Openness, even if a safety password is set.

Modifying F-parameters while safety password is set

F-parameters of a F-IO can only be modified if the safety password is not set.

Accessing TO objects in a S7 1200 CPU

The access to array tags for the TO objects TO_PositioningAxis and TO_CommandTable has been changed. You can find details in the chapter about S7-1200 Motion Control.

7.6 Major changes in V14 SP1

7.6.1 Major changes in V14 SP1

Introduction

The following changes were made in TIA Portal Openness API object modell V14 SP1, which may impact your existing applications:

Change	Required program code adjustment
Impoved handling for master copies	The CreateFrom action will create a new object based on a master copy in a library and place it in the composition where the action was called. The CreateFrom action only supports master copies containing only single objects. The return type corresponds to the respective composed type. The following composition support CreateFrom: • Siemens.Engineering.HW.DeviceComposition • Siemens.Engineering.HW.DeviceItemComposition • Siemens.Engineering.SW.Blocks.PlcBlockComposition • Siemens.Engineering.SW.Tags.PlcTagTableComposition • Siemens.Engineering.SW.Tags.PlcTagComposition • Siemens.Engineering.SW.Types.PlcTypeComposition • Siemens.Engineering.SW.TechnologicalObjects.TechnologicalInstanceDBComposition • Siemens.Engineering.SW.Tags.PlcUserConstantComposition • Siemens.Engineering.Hmi.Tag.TagTableComposition • Siemens.Engineering.Hmi.Tag.TagComposition • Siemens.Engineering.Hmi.Screen.ScreenComposition • Siemens.Engineering.Hmi.Screen.ScreenTemplateComposition • Siemens.Engineering.Hmi.RuntimeScripting.VBScriptComposition • Siemens.Engineering.HW.SubnetComposition • Siemens.Engineering.HW.DeviceUserGroupComposition • Siemens.Engineering.SW.Blocks.PlcBlockUserGroupComposition • Siemens.Engineering.SW.ExternalSources.PlcExternalSourceUserGroupComposition • Siemens.Engineering.SW.Tags.PlcTagTableUserGroupComposition • Siemens.Engineering.SW.Types.PlcTypeUserGroupComposition
Improved handling for global libraries	Existing actions on global libraries can now be modifying actions, e.g. delete a master copy from a global library. UpdateProject and UpdateLibrary do not longer use the UpdatePathsMode and DeleteUnusedVersionsMode parameters. Unused versions are not deleted after an update

Change	Required program code adjustment
Change System.String to System.IO.FileInfo Change System.String to System.IO.DirectoryInfo	All occurrences where a string path had to be specified are using FileInfo path or a DirectoryInfo path. For example: • Open project • Open library • Create project • Create global llibrary • ...

New items in the object model

Name	Type	Namespace	Comment
PlcUserConstant	Class	Siemens.Engineering.SW.Tags	Split from PlcConstant.
PlcUserConstantComposition	Class	Siemens.Engineering.SW.Tags	Split from PlcConstantComposition.
PlcSystemConstant	Class	Siemens.Engineering.SW.Tags	Split from PlcConstant.
PlcSystemConstantComposition	Class	Siemens.Engineering.SW.Tags	Split from PlcConstantComposition.
MultilingualTextItem	Class	Siemens.Engineering	Access to multilingual text.
MultilingualTextItemComposition	Class	Siemens.Engineering	Access to multilingual text.
TiaPortalTrustAuthority.FeatureTokens	Enum value	Siemens.Engineering	Access to TIA Portal settings.
TiaPortalSetting	Class	Siemens.Engineering.Settings	Access to TIA Portal settings.
TiaPortalSettingComposition	Class	Siemens.Engineering.Settings	Access to TIA Portal settings.
TiaPortalSettingsFolder	Class	Siemens.Engineering.Settings	Access to TIA Portal settings.
TiaPortalSettingsFolderComposition	Class	Siemens.Engineering.Settings	Access to TIA Portal settings.
LanguageAssociation	Class	Siemens.Engineering	Access to active languages.
LanguageComposition.Find	Method	Siemens.Engineering	Access to active languages.

Modified items in the object model

Name	Type	Namespace	Comment
PlcConstant	Class	Siemens.Engineering.SW.Tags	Published base class of PlcUserConstant and PlcSystemConstant.
PlcTag	Class	Siemens.Engineering.SW.Tags	Split from PlcConstantComposition.
ITargetComparable	Interface	Siemens.Engineering.Compare	String attribute DataTypeName instead of an open link DataType.
MultilingualText	Class	Siemens.Engineering	Access to multilingual text.

Name	Type	Namespace	Comment
ProjectComposition.Create	Method	Siemens.Engineering	Parameters changed to using a DirectoryInfo and string.
Project.Subnets	Attribute	Siemens.Engineering	Access to subnets
Project.Languages	Attribute	Siemens.Engineering	Moved to be an attribute of Siemens.Engineering.LanguageSettings to provide supported languages

Removed items in the object model

Name	Type	Namespace	Comment
PlcConstantComposition	Class	Siemens.Engineering.SW.Tags	Split in PlcSystemConstantComposition and PlcUserConstantComposition.
CompareResultElement.PathInformation	Attribute	Siemens.Engineering.SW.Tags	Not used anymore.
MultilingualText.GetText(CultureInfo cultureInfo)	Method	Siemens.Engineering.Compare	Modified concept for accessing text items of MultilingualText.
TiaPortalTrustAuthority.CustomerIdentification	Enum value	Siemens.Engineering	Not used anymore.
TiaPortalTrustAuthority.ElevatedAccessExtensions	Enum value	Siemens.Engineering	Not used anymore.

Behaviour changes

Name	Type	Namespace	Comment
PlcTag.Export(FileInfo path, ExportOptions options)	Method	Siemens.Engineering.SW.Tags	The value of the attribute LogicalAddress is always exported in international mnemonics now. German mnemonics are still accepted on import.
PlcTag.LogicalAddress	Attribute	Siemens.Engineering.SW.Tags	The value of the attribute LogicalAddress is always returned in international mnemonics now. German mnemonics are accepted on write.

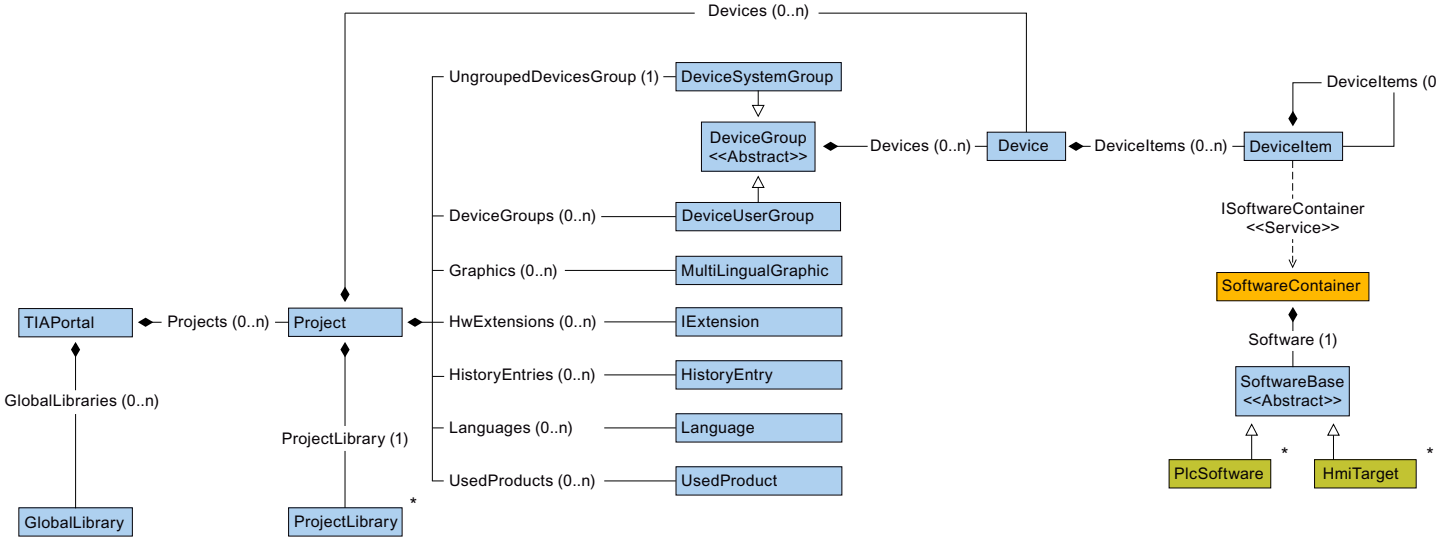
7.6.2 Major changes in the object model

Object model of TIA Portal Openness V14

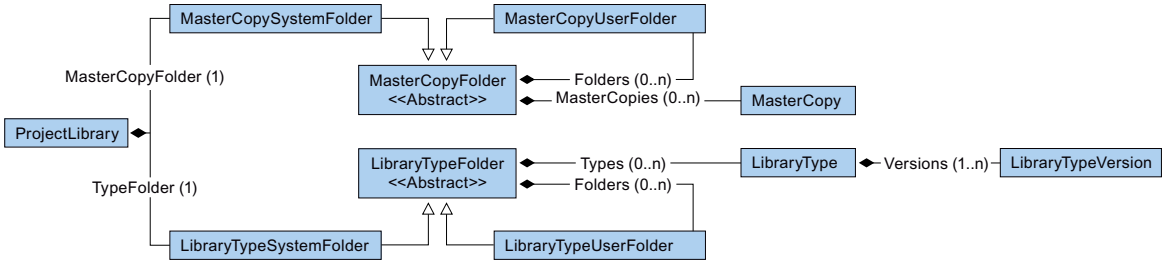
In order to allow you a comparison between the old and the new object model of TIA Portal Openness, the diagram below describes the object model of TIA Portal V14.

Note

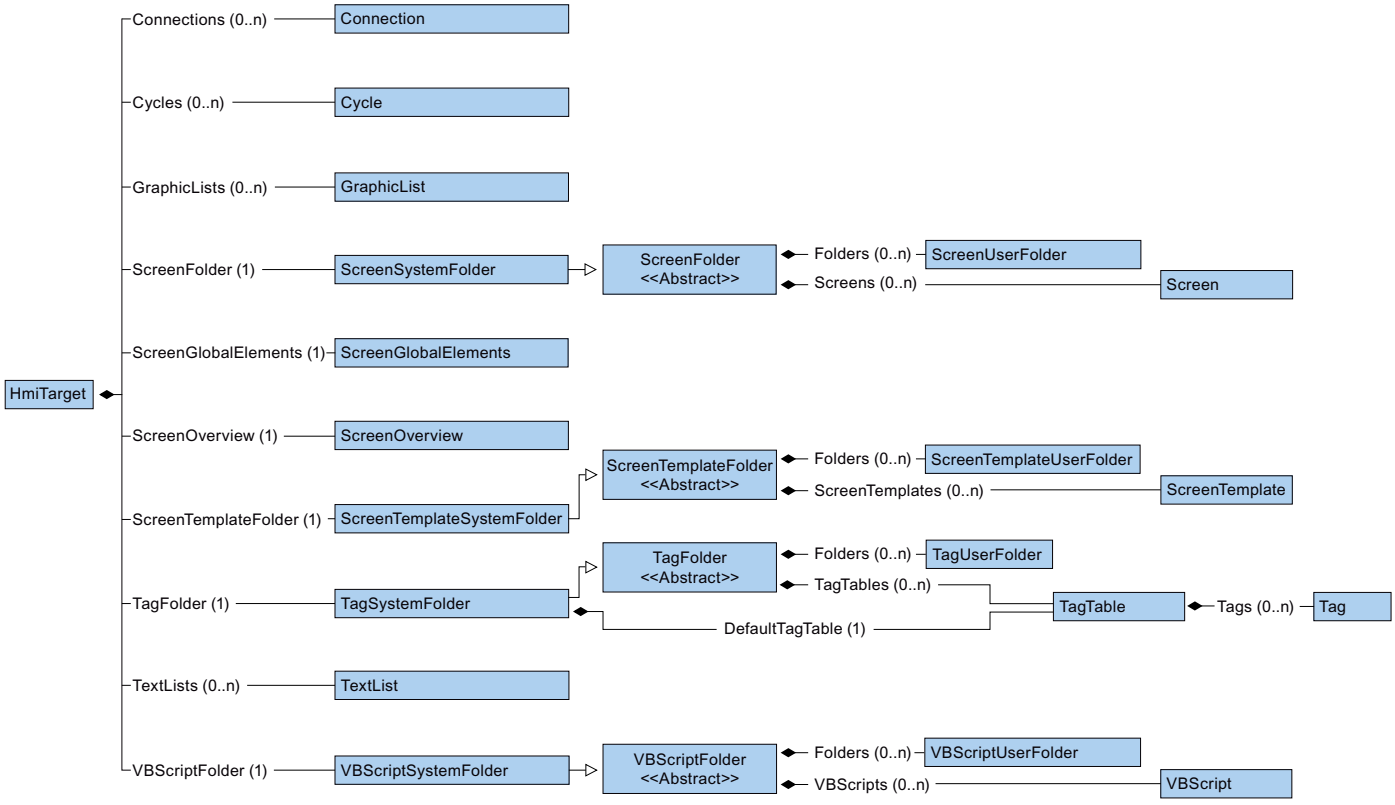
The object model described on the diagram is obsolete, for information about the object model of TIA Portal Openness V14 SP1 refer to TIA Portal Openness object model (Page 53)



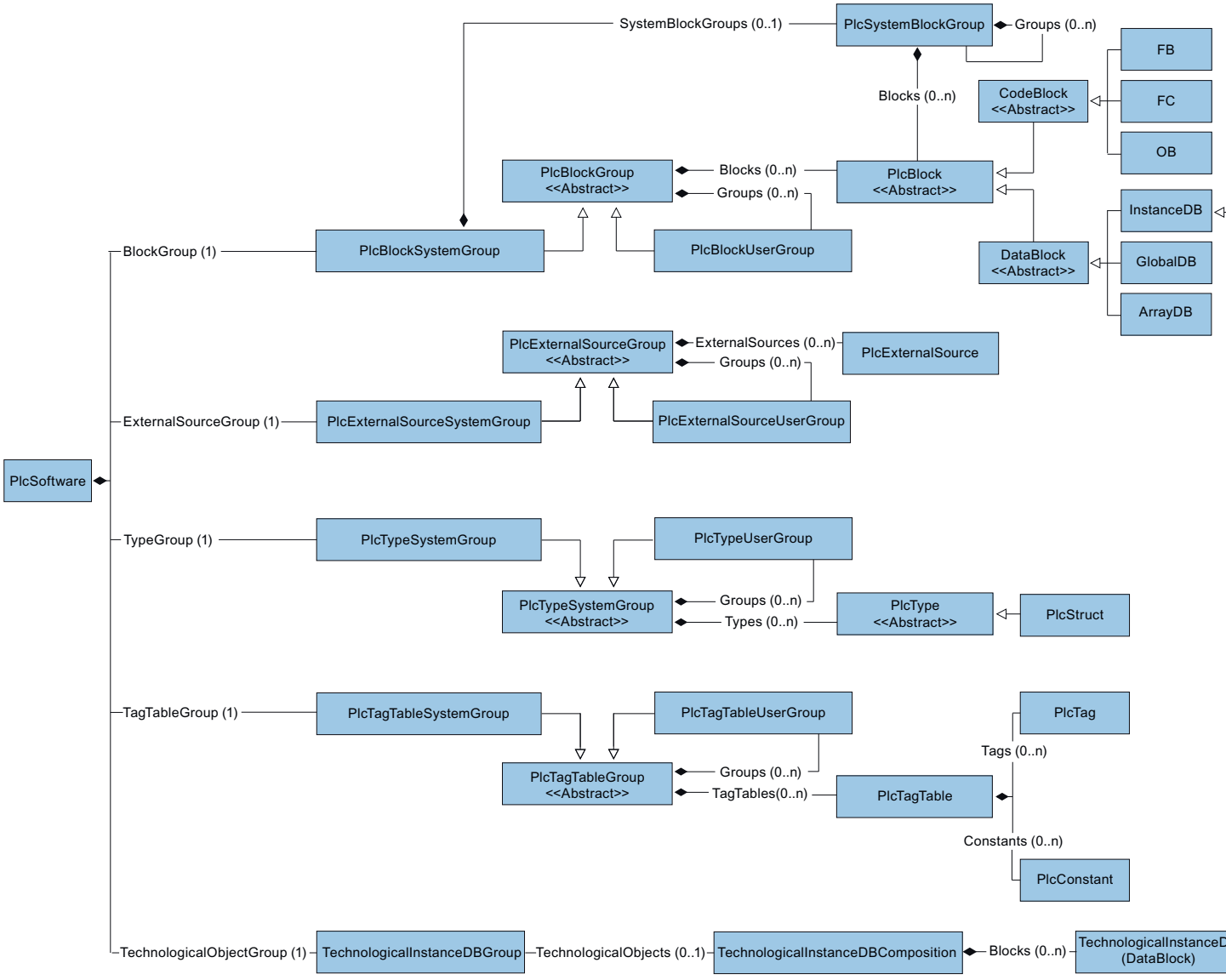
The following diagramm describes the objects which are located under ProjectLibrary.



The following diagramm describes the objects which are located under HmiTarget.

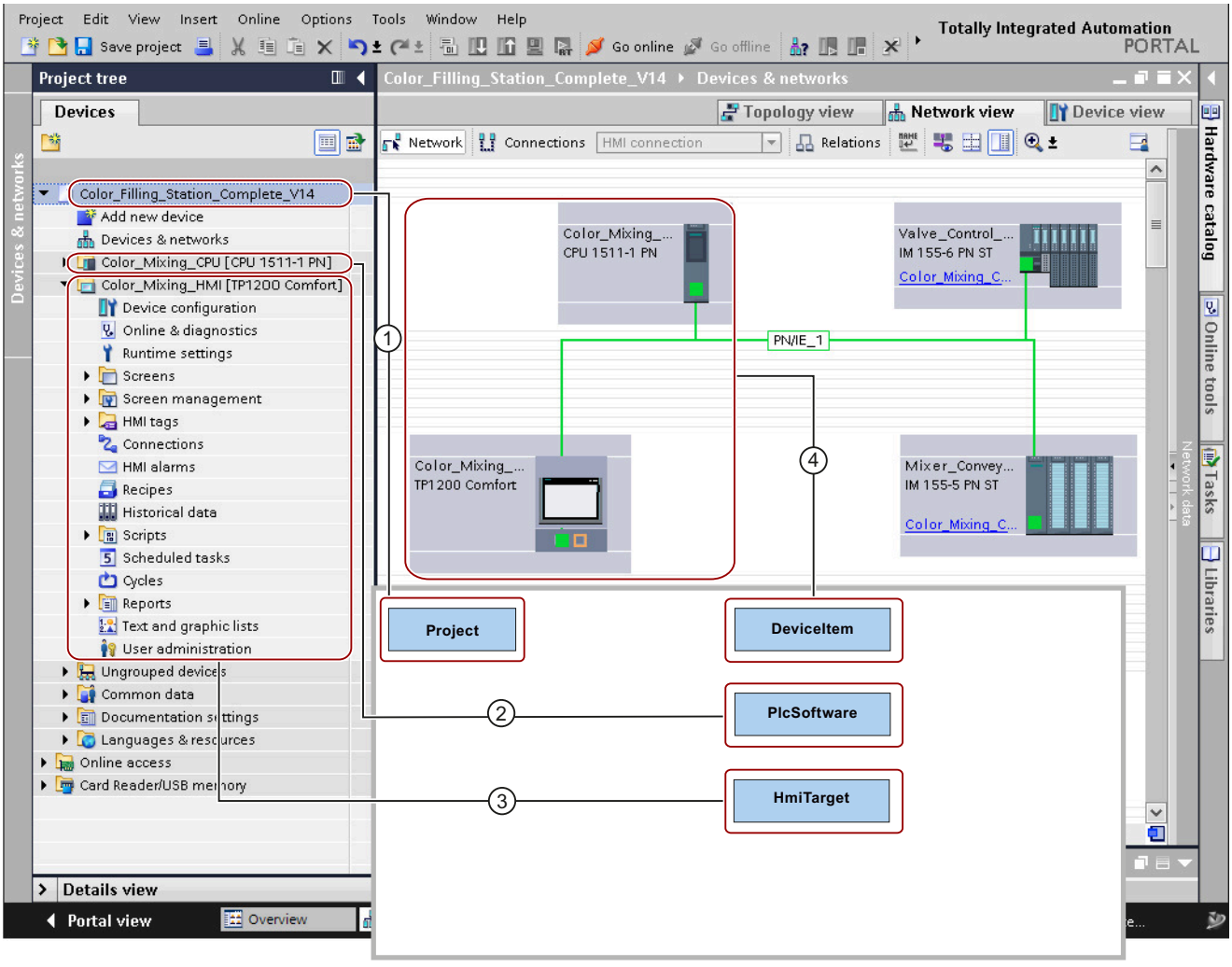


The following diagramm describes the objects which are located under PlcSoftware.


SystemBlockGroups (0..1)
PlcSystemBlockGroup
Groups (0..n)
FB
FC
OB
Blocks (0..n)
CodeBlock
<<Abstract>>
PlcBlockGroup
<<Abstract>>
Blocks (0..n)
Groups (0..n)
PlcBlock
<<Abstract>>
InstanceDB
GlobalDB
ArrayDB
DataBlock
<<Abstract>>
BlockGroup (1)
PlcBlockSystemGroup
PlcBlockUserGroup
PlcExternalSourceGroup
<<Abstract>>
ExternalSources (0..n)
Groups (0..n)
PlcExternalSource
ExternalSourceGroup (1)
PlcExternalSourceSystemGroup
PlcExternalSourceUserGroup
PlcSoftware
TypeGroup (1)
PlcTypeSystemGroup
PlcTypeUserGroup
PlcTypeSystemGroup
<<Abstract>>
Groups (0..n)
Types (0..n)
PlcType
<<Abstract>>
PlcStruct
TagTableGroup (1)
PlcTagTableSystemGroup
PlcTagTableUserGroup
PlcTag
Tags (0..n)
PlcTagTableGroup
<<Abstract>>
Groups (0..n)
TagTables(0..n)
PlcTagTable
Constants (0..n)
PlcConstant
TechnologicalObjectGroup (1)
TechnologicalInstanceDBGroup
TechnologicalObjects (0..1)
TechnologicalInstanceDBComposition
Blocks (0..n)
TechnologicalInstanceD
(DataBlock)

Relationship between TIA Portal and TIA Portal Openness object model

The figure below shows the relationship between the object model and a project in the TIA Portal:



① The object "Project" corresponds to an open project in the TIA Portal.

② The "PlcSoftware" object is of type "SoftwareBase"④, and corresponds to a PLC. The object's contents correspond to a PLC in the project navigation with access to objects such as blocks or PLC tags.

③ The "HmiTarget" object is of type "SoftwareBase"④, and corresponds to an HMI device. The object's contents correspond to an HMI device in the project navigation with access to objects such as screens or HMI tags.

④ The object "DeviceItem" corresponds to an object in the "Devices & Networks" editor. An object of the type "DeviceItem" can be a rack or an inserted module.

7.6.3 Changes on pilot functionality

Introduction

The following changes were made in API object modell V14 SP1 are only relevant for users, which have used the pilot functionality of HW Config in V14.

Modifications for TIA Portal Openness API types

TIA Portal Openness API type	new TIA Portal Openness API type
Siemens.Engineering.HW.IAddress	Siemens.Engineering.HW.Address
Siemens.Engineering.HW.IAddressController	Siemens.Engineering.HW.Features.AddressController
Siemens.Engineering.HW.IChannel	Siemens.Engineering.HW.Channel
Siemens.Engineering.HW.IDevice	Siemens.Engineering.HW.Device
Siemens.Engineering.HW.IDeviceItem	Siemens.Engineering.HW.DeviceItem
Siemens.Engineering.HW.IExtension	Siemens.Engineering.HW.Extensions
Siemens.Engineering.HW.IGsd	Siemens.Engineering.HW.Features.GsdObject
Siemens.Engineering.HW.IGsdDevice	Siemens.Engineering.HW.Features.GsdDevice
Siemens.Engineering.HW.IGsdDeviceItem	Siemens.Engineering.HW.Features.GsdDeviceItem
Siemens.Engineering.HW.IHardwareObject	Siemens.Engineering.HW.HardwareObject
Siemens.Engineering.HW.IHwIdentifier	Siemens.Engineering.HW.HwIdentifier
Siemens.Engineering.HW.IHwIdentifierController	Siemens.Engineering.HW.Features.HwIdentifierController
Siemens.Engineering.HW.IIoConnector	Siemens.Engineering.HW.IoConnector
Siemens.Engineering.HW.IIoController	Siemens.Engineering.HW.IoController
Siemens.Engineering.HW.IIoSystem	Siemens.Engineering.HW.IoSystem
Siemens.Engineering.HW.IInterface	Siemens.Engineering.HW.Features.NetworkInterface
Siemens.Engineering.HW.Extensions.ModuleInformationProvider	Siemens.Engineering.HW.Utilities.ModuleInformationProvider
Siemens.Engineering.HW.INode	Siemens.Engineering.HW.Node
Siemens.Engineering.HW.OPCUAExportProvider	Siemens.Engineering.HW.Utilities.OpcUaExportProvider
Siemens.Engineering.HW.IPort	Siemens.Engineering.HW.Features.NetworkPort
Siemens.Engineering.HW.IRole	Siemens.Engineering.HW.Features.HardwareFeature
	Siemens.Engineering.HW.Features.DeviceFeature
	Siemens.Engineering.HW.Utilities.ModuleInformationProvider
Siemens.Engineering.HW.SoftwareBase	Siemens.Engineering.HW.Software
Siemens.Engineering.HW.ISubnet	Siemens.Engineering.HW.Subnet
Siemens.Engineering.HW.ISoftwareContainer	Siemens.Engineering.HW.Features.SoftwareContainer
Siemens.Engineering.HW.ISubnetOwner	Siemens.Engineering.HW.Features.SubnetOwner

Modifications for Enums

TIA Portal Openness API type	Data type	new TIA Portal Openness API type	Data type
Siemens.Engineering.HW.Enums.AddressContext		Siemens.Engineering.HW.AddressContext	
Siemens.Engineering.HW.Enums.AddressIoType		Siemens.Engineering.HW.AddressIoType	
Siemens.Engineering.HW.Enums.AttachmentType		Siemens.Engineering.HW.MediumAttachmentType	
Siemens.Engineering.HW.Enums.BaudRate		Siemens.Engineering.HW.BaudRate	
Siemens.Engineering.HW.Enums.BusLoad		Siemens.Engineering.HW.CommunicationLoad	
Siemens.Engineering.HW.Enums.BusProfile		Siemens.Engineering.HW.BusProfile	
Siemens.Engineering.HW.Enums.CableLength		Siemens.Engineering.HW.CableLength	
Siemens.Engineering.HW.Enums.CableName	ulong	Siemens.Engineering.HW.CableName	long
Siemens.Engineering.HW.Enums.ChannelIoType	byte	Siemens.Engineering.HW.ChannelIoType	int
Siemens.Engineering.HW.Enums.ChannelType	byte	Siemens.Engineering.HW.ChannelType	int
Siemens.Engineering.HW.Enums.DeviceItemClassifications		Siemens.Engineering.HW.DeviceItemClassifications	
Siemens.Engineering.HW.Enums.InterfaceOperatingModes		Siemens.Engineering.HW.InterfaceOperatingModes	
Siemens.Engineering.HW.Enums.IpProtocolSelection		Siemens.Engineering.HW.IpProtocolSelection	
Siemens.Engineering.HW.Enums.MediaRedundancyRole		Siemens.Engineering.HW.MediaRedundancyRole	
Siemens.Engineering.HW.Enums.NetType		Siemens.Engineering.HW.NetType	
Siemens.Engineering.HW.Enums.ProfinetUpdateTimeMode		removed	
Siemens.Engineering.HW.Enums.RtClass	byte	Siemens.Engineering.HW.RtClass	int
Siemens.Engineering.HW.Enums.SignalDelaySelection	byte	Siemens.Engineering.HW.SignalDelaySelection	int
Siemens.Engineering.HW.Enums.SyncRole	byte	Siemens.Engineering.HW.SyncRole	int
Siemens.Engineering.HW.Enums.TransmissionRateAndDuplex	uint	Siemens.Engineering.HW.TransmissionRateAndDuplex	int

Modifications for attributes values of Siemens.Engineering.HW.IoConnector

Attribut	Data type	new name	Data type
ProfinetUpdateTimeMode	ProfinetUpdateTimeMode	PnUpdateTimeAutoCalculation	bool
ProfinetUpdateTime		PnUpdateTime	
AdaptUpdateTime		PnUpdateTimeAdaption	
WatchdogFactor		PNWatchdogFactor	
		DeviceNumber	string

Modifications for attributes values of Siemens.Engineering.HW.IoController

Attribut	Data type	new name	Data type
		DeviceNumber	string

Modifications for attributes values of Siemens.Engineering.HW.Node

Attribut	Data type	new name	Data type
HighestAddress		removed, available only on subnet	
TransmissionSpeed		removed, available only on subnet	
IsoProtocolUsed		UseIsoProtocol	
IpProtocolUsed		UseIpProtocol	
RouterAddressUsed		UseRouter	
PnDeviceNameAutoGener-ated		PnDeviceNameAutoGeneration	
DeviceNumber		removed, moved to IoConnector / IoController	

Modifications for attributes values of Siemens.Engineering.HW.Subnet

Attribut	Data type	new name	Data type
HighestAddress	byte	HighestAddress	int
CableConfiguration		PbCableConfiguration	
RepeaterCount		PbRepeaterCount	
CopperCableLength		PbCopperCableLength	
OpticalComponentCount		PbOpticalComponentCount	
OpticalCableLength		PbOpticalCableLength	
OpticalRingEnabled		PbOpticalRing	
OlmP12		PbOlmP12	
OlmG12		PbOlmP12	
OlmG12Eec		PbOlmG12Eec	
OlmG121300		PbOlmG121300	
AdditionalNetworkDevices		PbAdditionalNetworkDevices	
AdditionalDpMaster	byte	PbAdditionalDpMaster	int
TotalDpMaster	byte	PbTotalDpMaster	int
AdditionalPassiveDevice	byte	PbAdditionalPassiveDevice	int
TotalPassiveDevice	byte	PbTotalPassiveDevice	int
AdditionalActiveDevice	byte	PbAdditionalActiveDevice	int
TotalActiveDevice	byte	PbTotalActiveDevice	int
PbCommunicationLoad	BusLoad	PbAdditionalCommunicationLoad	CommunicationLoad
OptimizeDde		PbDirectDateExchange	
MinimizeTslot		PbMinimizeTslotForSlaveFailure	
OptimizeCableConfig		PbOptimizeCableConfiguration	
CyclicDistribution		PbCyclicDistribution	
TslotInit		PbTslotInit	
Tslot		PbTslot	
MinTsdr		PbMinTsdr	
MaxTsdr		PbMaxTsdr	
Tid1		PbTid1	
Tid2		PbTid2	

Attribut	Data type	new name	Data type
Trdy		PbTrdy	
Tset		PbTset	
Tqui		PbTqui	
Ttr		PbTtr	
TtrMs		removed	
TtrTypical		PbTtrTypical	
TtrTypicalMs		removed	
Watchdog		PbWatchdog	
WatchdogMs		removed	
Gap	byte	PbGapFactor	int
RetryLimit	byte	PbRetryLimit	int
IsochronMode		IsochronousMode	
AdditionalDevice		PbAdditionalPassivDeviceForIsochro-nousMode	
TotalDevice		PbTotalPassivDeviceForIsochronous-Mode	
DpCycleTimeAutoCalc		DpCycleMinTimeAutoCalculation	
TiToAutoCalc		IsochronousTiToAutoCalculation	
Ti		IsochronousTi	
To		IsochronousTo	

Modifications for attributes values of Siemens.Engineering.Project

Attribut	Data type	new name	Data type
.HwExtensions		.HwUtilities	

Modifications for attributes values of Siemens.Engineering.HW.Baudrate

Attribut	Data type	new name	Data type
BaudRate.BAUD_9600		BaudRate.BAUD9600	
BaudRate.BAUD_19200		BaudRate.BAUD19200	
BaudRate.BAUD_45450		BaudRate.BAUD45450	
BaudRate.BAUD_93750		BaudRate.BAUD93750	
BaudRate.BAUD_187500		BaudRate.BAUD187500	
BaudRate.BAUD_500000		BaudRate.BAUD500000	
BaudRate.BAUD_1500000		BaudRate.BAUD1500000	
BaudRate.BAUD_3000000		BaudRate.BAUD3000000	
BaudRate.BAUD_6000000		BaudRate.BAUD6000000	
BaudRate.BAUD_12000000		BaudRate.BAUD12000000	

Modifications for attributes values of Siemens.Engineering.HW.CableLength

Attribut	Data type	new name	Data type
CableLength.Unknown		CableLength.None	
CableLength.Length_20m		CableLength.Length20m	
CableLength.Length_50m		CableLength.Length50m	
CableLength.Length_100m		CableLength.Length100m	
CableLength.Length_1000m		CableLength.Length1000m	
CableLength.Length_3000m		CableLength.Length3000m	

Modifications for attributes values of Siemens.Engineering.HW.ChannelIoType

Attribut	Data type	new name	Data type
ChannelIoType.Unknown		ChannelIoType.Complex	

Modifications for attributes values of Siemens.Engineering.HW.IpProtocolSelection

Attribut	Data type	new name	Data type
IpProtocolSelection.AddressTailoring		IpProtocolSelection.ViaIoController	

Modifications for attributes values of Siemens.Engineering.HW.TransmissionRateAndDuplex

Attribut	Data type	new name	Data type
TransmissionRateAndDuplex.Unknown		TransmissionRateAndDuplex.None	
TransmissionRateAndDuplex.TP10Mbps_HalfDuplex		TransmissionRateAndDuplex.TP10MbpsHalfDuplex	
TransmissionRateAndDuplex.TP10Mbps_FullDuplex		TransmissionRateAndDuplex.TP10MbpsFullDuplex	
TransmissionRateAndDuplex.AsyncFiber10Mbps_HalfDuplex		TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex	
TransmissionRateAndDuplex.AsyncFiber10Mbps_FullDuplex		TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex	
TransmissionRateAndDuplex.TP100Mbps_HalfDuplex		TransmissionRateAndDuplex.TP100MbpsHalfDuplex	
TransmissionRateAndDuplex.TP100Mbps_FullDuplex		TransmissionRateAndDuplex.TP100MbpsFullDuplex	
TransmissionRateAndDuplex.FO100Mbps_FullDuplex		TransmissionRateAndDuplex.FO100MbpsFullDuplex	
TransmissionRateAndDuplex.X1000Mbps_FullDuplex		TransmissionRateAndDuplex.X1000MbpsFullDuplex	
TransmissionRateAndDuplex.FO1000Mbps_FullDuplex_LD		TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD	
TransmissionRateAndDuplex.FO1000Mbps_FullDuplex		TransmissionRateAndDuplex.FO1000MbpsFullDuplex	

Attribut	Data type	new name	Data type
TransmissionRateAndDuplex.TP1000Mbps_FullDuplex		TransmissionRateAndDuplex.TP1000MbpsFullDuplex	
TransmissionRateAndDuplex.FO10000Mbps_FullDuplex		TransmissionRateAndDuplex.FO10000MbpsFullDuplex	
TransmissionRateAndDuplex.FO100Mbps_FullDuplex_LD		TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	
TransmissionRateAndDuplex.POFPCF100Mbps_FullDuplex_LD		TransmissionRateAndDuplex.POFPCF100MbpsFullDuplexLD	

7.6.4 Changes for export and import

7.6.4.1 Changes for export and import

Introduction

The export and import via TIA Portal Openness API was extended in V14 SP1 in order to handle comments at array elements. This required a new schema. Block interface import and export will handle from now on two schema versions:

- For import: The decision about used schema version is made based on namespace: <Sections xmlns=http://www.siemens.com/automation/Openness/SW/Interface/v2>
- For export: The decision about used schema version is made based on the project version. Projects V14 SP1 lead to version 2, projects V14 lead to version v1

7.6.4.2 Changes in API

Generate source

The following methods have been removed from ProgramBlocks:

- GenerateSourceFromBlocks
- GenerateSourceFromTypes

The following methodes have been added:

- GenerateSource to PlcExternalSourceSystemGroup

Example

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
using System.Security;
namespace ChangesInTheAPI
{
internal class Program
{
private static void Main(string[] args)
{
// generate source for V14
var blocks = new List<PlcBlock>() { block1 };
var types = new List<PlcBlock>() { udt1 };
var fileInfoBlock = new FileInfo(@"D:\Export\Block.scl");
var fileInfoType = new FileInfo(@"D:\Export\Type.udt");
PlcBlockSystemGroup blocksGroup = ...;
blocksGroup.GenerateSourceFromBlocks(blocks, fileInfo);
PlcTypeSystemGroup plcDataTypesGroup = ...;
plcDataTypesGroup.GenerateSourceFromTypes(types, fileInfo);
//generate source as of V14 SP1
var blocks = new List<PlcBlock>(){ block1 };
var types = new List<PlcBlock>() { udt1 };
var fileInfoBlock = new FileInfo(@"D:\Export\Blocks.scl");
var fileInfoType = new FileInfo(@"D:\Export\Type.udt");
PlcExternalSourceSystemGroup externalSourceGroup = plc.ExternalSourceGroup;
externalSourceGroup.GenerateSource(blocks, fileInfoBlock);
externalSourceGroup.GenerateSource(types, fileInfoType);
}
}
}
```

7.6.4.3 Schema extension

Schema extension for comments and start values

Comments and start values are stored in new element called "Subelement" which refers to the array element with "Path" attribute.

Subelement contains start value and comment for the referenced array element. Attribute "Path" at StartValue is removed in the new schema.

Schema definition of "Subelement":

```
<xs:element name="Subelement" type="Subelement_T"/>
  <xs:complexType name="Subelement_T">
      <xs:sequence>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element ref="StartValue"/>
              <xs:element ref="Comment"/>
          </xs:choice>
      </xs:sequence>
      <xs:attribute name="Path" type="IndexPath_TP"/>
  </xs:complexType>
```

Extension of member type:

```
<xs:complexType name="Member_T">
       <xs:sequence>
           <xs:element ref="AttributeList" minOccurs="0" maxOccurs="1"/>
           <xs:choice minOccurs="0" maxOccurs="unbounded">
               <xs:element ref="Member"/>
               <xs:element ref="Sections"/>
               <xs:element ref="StartValue"/>
               <xs:element ref="Comment"/>
               <xs:element ref="Subelement"/>
           </xs:choice>
       </xs:sequence>
</xs:complexType>
```

Examples:

Storage of comments and start values in simple arrays:

```
<Member Name="Static_1" Datatype="Array[0..1] of Bool">
    <Comment>
        <MultiLanguageText Lang="de-DE">comment for array</MultiLanguageText>
    </Comment>
    <Subelement Path="0">
        <StartValue>true</StartValue>
        <Comment>
            <MultiLanguageText Lang="de-DE">comment for array element 0</MultiLanguageText>
        </Comment>
    </Subelement>
    <Subelement Path="1">
        <StartValue>true</StartValue>
        <Comment>
            <MultiLanguageText Lang="de-DE">comment for array element 1</MultiLanguageText>
        </Comment>
    </Subelement>
</Member>
```

Storage of comments and start values in arrays of UDT:

```
<Member Name="Static_1" Datatype="Array[0..1] of &quot;User_data_type_1&quot;">
    <Comment>
        <MultiLanguageText Lang="de-DE">comment for array</MultiLanguageText>
    </Comment>
    <Subelement Path="0">
        <Comment>
            <MultiLanguageText Lang="de-DE">cmt array  0</MultiLanguageText>
        </Comment>
    </Subelement>
    <Sections>
        <Section Name="None">
            <Member Name="Element_1" Datatype="Bool">
                <Subelement Path="0">
                    <StartValue>true</StartValue>
                    <Comment>
                        <MultiLanguageText Lang="de-DE">comment for element 0</MultiLanguageText>
                    </Comment>
                </Subelement>
                <Subelement Path="1">
                    <StartValue>true</StartValue>
                    <Comment>
                        <MultiLanguageText Lang="de-DE">comment for element 1</MultiLanguageText>
                    </Comment>
                </Subelement>
            </Member>
            <Member Name="Element_2" Datatype="Struct">
                <Member Name="Element_1" Datatype="Int">
                    <Subelement Path="0">
                        <StartValue>11</StartValue>
                        <Comment>
                            <MultiLanguageText Lang="de-DE">comment for element 0</MultiLanguageText>
                        </Comment>
                    </Subelement>
                </Member>
            </Member>
        </Section>
    </Sections>
</Member>
```

Storage of comments and start values in arrays of struct:

```
<Member Name="Static_1" Datatype="Array[0..1] of Struct">
    <Member Name="Static_1" Datatype="Int">
        <Subelement Path="0">
            <StartValue>11</StartValue>
            <Comment>
                <MultiLanguageText Lang="de-DE">comment for int elem</MultiLanguageText>
            </Comment>
        </Subelement>
    </Member>
    <Member Name="Static_2" Datatype="Bool">
        <Subelement Path="1">
            <StartValue>true</StartValue>
            <Comment>
                <MultiLanguageText Lang="de-DE">comment for bool elem</MultiLanguageText>
            </Comment>
        </Subelement>
    </Member>
</Member>
```

7.6.4.4 Schema changes

Access node in SW.PlcBlocks.Access.xsd

Type attribute of the Access node has been moved to the children nodes of Access at

- AbsoluteOffset as required
- Address as optional

```
<StlStatement UId="22">
  <StlToken Text="L" />
  <Access Scope="Address">
    <Address Area="Local" Type="Word" BitOffset="80" />
  </Access>
</StlStatement>
```

Type attribute of Constant has been replaced with new ConstantType subnode.

```
<Access Scope="LocalConstant">
  <IntegerAttribute Name="NumBLs" Informative="true">5</IntegerAttribute>
  <Constant Name="LocalConstant_A">
    <ConstantType Informative="true">Int</ConstantType>
    <ConstantValue Informative="true">10</ConstantValue>
    <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute>
  </Constant>
</Access>
```

The value of the Scope attribute in Access has been renamed to TypedConstant if the ConstantValue contains type qualified value(e.g.: int#10).

Constant does not have Type attribute if ConstantValue contains type qualified value (e.g.: int#10).

Local variables do not have Address node if Scope is LocalVariable.

If an Access is nested within another Access at any level, only the outer Access must have an UId.

Address node in SW.PlcBlocks.Access.xsd

BitOffset attribute of Address node became optional.

Declarations for exporting absolute access have changed as shown in the following table:

Area as of V14 SP1	Type	Block number	Bit offset	Example
DB	Block_DB	mandatory	forbidden	OPN %DB12
DB	unordered	existing	mandatory	%DB100.DBX10.3
DB	unordered	not existing	mandatory	%DB100.DBX10.3
L	unordered	forbidden	mandatory	%LW10.0
I Q M	unordered	forbidden	mandatory	%I0.0 %Q0.0 %M0.0
T C	unordered	forbidden	mandatory	%T0 %C1
Block_FC Block_FB	Block_FC Block_FB	mandatory	forbidden	Call %FB4, %DB5 Input_1 := %FC10 Call %FB4, %DB5 Input_2 := %FB11
PeripheryInput	unordered	forbidden	mandatory	
Periphery Output	unordered	forbidden	mandatory	

Area node in SW.PlcBlocks.Access.xsd

Area node has got a simplified enum list:

- LocalC and LocalN became Local
- DBc, DBv, DBr are eliminated.

CallInfo node in SW.PlcBlocks.Access.xsd

Name attribute of CallInfo node became optional

BlockType attribute of CallInfo node became required

+2.2.5 User Block Calls

Constant node in SW.PlcBlocks.Access.xsd

Constant node references CostantType node with minOccurs=0

Constant node doesn't reference IntegerAttribute node any more

ConstantValue node in SW.PlcBlocks.Access.xsd

ConstantValue node gets an Informative attribute

Instruction node in SW.PlcBlocks.Access.xsd

Instruction node references Acces node with minOccurs=0

Parameter attributes Section, Type and TemplateReference have been deleted at Instruction.

Parameter node in SW.PlcBlocks.Access.xsd

SectionName attribute of the Parameter node became optional.

Values for Scope in SW.PlcBlocks.Access.xsd

Enum list of Scope has been extended with:

- TypedConstant
- AddressConstant
- LiteralConstant
- AlarmConstant
- Address
- Statusword
- Expression
- Call
- CallWithType

Statusword node in SW.PlcBlocks.Access.xsd

Enum list of Statusword has been extended with:

- STW

ConstantType node in SW.PlcBlocks.Access.xsd

New node ConstantType is introduced with optionally used attribute Informative.

CallRef node in SW.PlcBlocks.LADFBD.xsd

CallRef node is renamed to Call and omits the BooleanAttribute subnode.

InstructionRef node in SW.PlcBlocks.LADFBD.xsd

InstructionRef node is replaced by Part node

Part node in SW.PlcBlocks.LADFBD.xsd

New node ConstantType is introduced and replaces the InstructionRef node

- Attributes: Name and Version
- Subnodes: Instruction subnode as new choice to existing Equation
- doesn't have neither BooleanAttribute subnode nor Gate attribute

Wire node in SW.PlcBlocks.LADFBD.xsd

Name attribe of Wire node removed.

TemplateReference node in SW.PlcBlocks.LADFBD.xsd

TemplateReference node is deleted.

StatementList node in SW.PlcBlocks.STL.xsd

Enum list of of StatementList (STL_TE):

- L_STW has been removed
- T_STW has been removed

7.6.4.5 Behaviour changes

Absolute Access

In V14 the import of absolute access has been aborted for most combinations. As of V14 SP1 the import of absolute access works for the following areas:

- Input
- Outpt
- Memory
- Timer, if supported on the PLC
- Counter, if supported on the PLC
- DB
- DI

If a symbol access and an absolute access is used at the same time and is not rejected by schema or node kind validation the import will only succeed when box access informations are successfully resolved. When the symbol access leads to different information in comparison to the absolute access the import is rejected.

```
<Access Scope="Address">
    <Address Area="Memory" Type="Word" BitOffset="0" />
</Access>
```

```
<CallInfo Name="Block_1" BlockType="FC">
    <Parameter Name="Input_1" Section="Input" Type="Int" />
    <Parameter Name="Input_1" Section="Input" Type="Int" />
<!-- Import will be aborted because parameter name 'Input_1' is used more than once -->
    <Parameter Name="Output_1" Section="NotExisting" Type="Int" />
<!-- Import will be aborted because the section 'NotExisting' can not be used. -->
    <Parameter Name="ENO" Section="Output" Type="Int" />
<!-- Import will be aborted because the parameter name 'ENO' is restricted and can not be used. -->
</CallInfo>
```

Indirect DB Access

As of V14 SP1 indirect DB Access can only be imported, when the 'offset', 'type' and 'symbol' are provided.

```
...
    <Access Scope="LocalVariable" UId="21">
        <Symbol>
            <Component Name="Output_3" />
            <AbsoluteOffset BitOffset="16" Type="Word" />
        </Symbol>
    </Access>
...
```

Symbolic and absolute information for local access

When importing "symbolic access" all possible provided "absolute access informations" are validated, if they are not flaged as "informative". As of V14 SP1 the import will be aborted when the absolute information does not match.

Block interface constraints

In V14SP1 several constrains are checked. These constrains are well known to users of the block interface editor. Whenever the block interface editor renames a parameter by adding or increasing '_1' the OPNS import will be aborted.

The following constrains are validated for instance:

- Duplicated parameter names
- Wrong section names. Including 'Return-Section' for FB blocks
- Restricted words

Sorting sections on import

When the called block does not exist at the time of import the interface definition at the call side will be used to display the called user block. In V14 SP1 the sections will be sorted in the order they would be displayed in the block interface of the called block, if it would have been existing with the same parameters.

The section order of the parameters imported is:

- Input
- Output

The following STL xml example

```
<StlStatement UId="21">
<StlToken Text="CALL" />
    <Access Scope="Call">
        <CallInfo Name="Block_2" BlockType="FC">
            <Parameter Name="Output_1" Section="Output" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_3" />
                    </Symbol>
                </Access>
            </Parameter>
            <Parameter Name="Input_1" Section="Input" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_1" />
                    </Symbol>
                </Access>
            </Parameter>
            <Parameter Name="Output_2" Section="Output" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_4" />
                    </Symbol>
                </Access>
            </Parameter>
            <Parameter Name="Input_2" Section="Input" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_2" />
                    </Symbol>
                </Access>
            </Parameter>
        </CallInfo>
    </Access>
</StlStatement>
```

will result in

```
CALL  "Block_2"
    Input_1  :="Tag_1"
    Input_2  :="Tag_2"
    Output_1 :="Tag_3"
    Output_2 :="Tag_4"
```

Unique user block callee names

In TIA Portal names have to be unique. This means for instance a tag can not have the same name like a block. For TIA Portal Openness API XML import this means when the XML contains a user block call, where the called block does not exist at the time of import, the name of the called block has to be unique to all existing names in the project. When the called block name is not unique the import will be aborted.

In the following example the import will be aborted, because the Name of the called Block "Tag_1" is already used for a tag table.

```
...
    <SW.Tags.PlcTag ID="1" CompositionName="Tags">
        <AttributeList>
            <DataTypeName>Int</DataTypeName>
            <LogicalAddress>%MW2</LogicalAddress>
            <Name>Tag_1</Name>
        </AttributeList>
    </SW.Tags.PlcTag>
...
...
    <StlStatement UId="21">
        <StlToken Text="CALL" />
        <Access Scope="Call">
            <CallInfo Name="Tag_1" BlockType="FC">
                <Parameter Name="Input_1" Section="Input" Type="Int">
                    <Access Scope="GlobalVariable">
                        <Symbol>
                            <Component Name="Tag_1" />
                        </Symbol>
                    </Access>
                </Parameter>
...
```

In the following example the import will be aborted, because two parameters have the same name ""Input1".

```
<StlStatement UId="22">
    <StlToken Text="CALL" />
    <Access Scope="Call">
        <CallInfo Name="Block_1" BlockType="FB">
            <Instance Scope="GlobalVariable">
                <Component Name="Block_1_DB" />
            </Instance>
            <Parameter Name="Input1" Section="Input" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_9" />
                    </Symbol>
                </Access>
            </Parameter>
            <Parameter Name="Input1" Section="Input" Type="Time">
                <Access Scope="TypedConstant">
                    <Constant>
                        <ConstantValue>T#1s</ConstantValue>
                    </Constant>
                </Access>
            </Parameter>
        </CallInfo>
    </Access>
</StlStatement>
```

Library block calls

The imported XML may contain calls of user blocks. These user blocks are identified by name.

User blocks can also call library elements. These library elements can be generated as 'library block calls'. Because library blocks are using the same namespace as user blocks, the import of a user block call done by name can call the implementation of a library block.

Before V14 SP1 the import tried to map the parameters between the user block call and the instruction block call. Sometimes the import aborted, sometimes the import deleted all not matching parameters.

As of V14 SP1 the user block call will still find the library block, but the call will not become valid.

Block type mismatch

When the XML contains a user block call of 'Block_1' with more parameters as the corresponding FC in the project as of V14 SP1 the import defines a new called block interface matching the user block call from the XML. The next program block compile will attempt the call update.

New scopes for contants

With V14SP1 several new scopes for constants have been invented. The import only succeeds when the values in the xml match the constant scope. The import may abort when not all the provided information for a constant match the existing constant.

```
...
    <Access Scope="LiteralConstant">
        <Constant>
            <ConstantType>Int</ConstantType>
            <ConstantValue>16#0000_0001</ConstantValue>
        </Constant>
    </Access>
...
    <Access Scope="TypedConstant">
        <Constant>
            <ConstantValue>Int#10</ConstantValue>
        </Constant>
    </Access>
...
    <Access Scope="LiteralConstant">
        <Constant>
            <ConstantType>Int</ConstantType>
            <ConstantValue>10</ConstantValue>
        </Constant>
    </Access>
...
    <Access Scope="GlobalConstant">
        <Constant Name="Constant_1" />
    </Access>
...
    <Access Scope="LocalConstant">
        <Constant Name="Constant_1" />
    </Access>
...
    <Access Scope="AddressConstant">
        <Constant Name="Tag_1" />
    </Access>
...
    <Access Scope="AlarmConstant">
        <Constant>
            <ConstantType>C_Alarm</ConstantType>
            <ConstantValue>16#0000_0001</ConstantValue>
        </Constant>
    </Access>
...
```

Instruction version annotation

As of V14 SP1 only instruction versions usable on the PLC to import to can be imported.When no instruction version is annotated in the xml the version selected in the PLC will be used. In LAD and FBD some elements represented as instruction do not use versioning. These elements can only be imported without version.

```
...
    <Part Name="MIN" Version="1.0" UId="27" DisabledENO="false">
        <TemplateValue Name="card" Type="Cardinality">2</TemplateValue>
        <TemplateValue Name="value_type" Type="Type">Int</TemplateValue>
    </Part>
    <Part Name="MIN" UId="28" DisabledENO="false">
        <TemplateValue Name="card" Type="Cardinality">2</TemplateValue>
        <TemplateValue Name="value_type" Type="Type">Int</TemplateValue>
    </Part>
...
```

Disabled ENO

The "disabled ENO" feature is used on 1200 and 1500 PLC to deactivate runtime consuming ENO connection state calculation.

As of V14 SP1 the DisabledENO flag can only be imported on PLC's supporting the feature.

```
...
    <Part Name="Add" UId="24" DisabledENO="false">
        <TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
        <TemplateValue Name="SrcType" Type="Type">Int</TemplateValue>
    </Part>
...
```

Type validation for absolute L-Stack access

As of V14 SP1 the import is aborted when the type can not be used or mapped.

Validation of index idents

Index access's are usable where 'symbolic access to memory' is defined. For instance, local access, global access, indirect access.

When a literal constant is used as index, the singed and unsigned integer types are changed to Dint. As of V14 SP1 the import is aborted when a type outside the mentioned range is provided.

All index access's are checked, whether the kind of access can be used as 'index access' at all. As of V14 SP1 the import is aborted when the defined index access can not be used.

Element order sorting

As of V14 SP1 the elements in LAD and FBD will be sorted 'code generation order' where automatically possible during export. In some very rare cases the exported XML is not importable again. In these cases either the XML has to be adapted or the corresponding networks have to be deleted and reprogrammed. But the order of wires and references is still not reliable.

Alarm Constants

With V14 SP1 the compile checks for valid alarm constants have been extended. It might occur that projects are not compilable in V14 SP1 due to an xml imported in V14 with broken alarm constants. In this case open the relevant network in LAD/FDB editor and delete the alarm actual operand. The editor will automatically recreate a valid alarm constant.

```
<FlgNet>
    <Parts>
        <Access Scope="AlarmConstant" UId="21">
            <Constant>
                <ConstantType>C_Alarm</ConstantType>
                <ConstantValue>16#0000_0002</ConstantValue>
            </Constant>
        </Access>
        <Call UId="22">
            <CallInfo Name="Block_1" BlockType="FB">
                <Instance UId="23" Scope="GlobalVariable">
                    <Component Name="Block_1_DB" />
                </Instance>
                <Parameter Name="Input_1" Section="Input" Type="C_Alarm" />
            </CallInfo>
        </Call>
    </Parts>
    <Wires>
        <Wire UId="24">
            <Powerrail />
            <NameCon UId="22" Name="en" />
        </Wire>
        <Wire UId="25">
            <IdentCon UId="21" />
            <NameCon UId="22" Name="Input_1" />
        </Wire>
    </Wires>
</FlgNet>
```

Constraints for instances of user blocks and instructions

In V14 it was possible to import user FC block calls with an instance and sometimes even compile these calls.

As of V14 SP1 the import of instances is only possible where instances are supported. Existing projects with instances at FC user block calls and instructions may not compile any more. In this case the call has to be deleted and reprogrammed. Any attempt to do a call update or 'other means of automated repair' will fail.

EnEno visible

In V14 the EN and ENO connections of 'InstructionRef' have been usable or not depending on the ENENO flag.

As of V14SP1 the OPNS during the import based on the element and the wiring either the EN and ENO connections are used. Because of this automatic detection a different EN and ENO connection usage can be noticed. Most probably only the IEC timer and IEC counter boxes may show some issues.

UId assignment

The assignment of UIds to parts, access and wires changes with V14 SP1. The UIds for statements, CallInfo and operands have to be unique within a compile unit. From TIA Portal point of view the UIds in the XML are keys, without any additional meaning besides identifying an element.

Checking of character strings

More strict checks concerning quotation marks, surrogate characters and control characters are performed for the Name attribute during the import of

- IntegerAttribute
- StringAttribute
- DateAttribute
- AutomaticTyped
- Component
- Invisible
- Label
- NameCon
- Negated
- TemplateValue
- CallInfo
- Instruction
- Parameter
- Part
- Step

More strict checks concerning surrogate characters and control characters are performed during the import of

- Titles of blocks and networks
- LineComment text
- Constant strings (String, WString, Char, Wchar typed)

More strict checks concerning surrogate characters and control characters (tab and new line allowed) are performed during the import of

- Comments of blocks and networks
- String attributes
- Nodes defining multilanguage texts, e.g. Alarmtext, Comments
- Token texts

Case insensitivity of template operations and parameters

As of V14 SP1 case insenitivity of template operations for instructions and call or instruction parameters will be imported and automatically corrected.

The following code will be imported and the incorrect value "Eq" will be corrected to "EQ" and the incorrect parameter "iN1" will be corrected to "IN1":

```
<StlStatement UId="22">
    <StlToken Text="CALL" />
    <Access Scope="Call">
        <Instruction Name="CompType">
            <TemplateValue Name="src_type" Type="Type">Variant</TemplateValue>
            <TemplateValue Name="relation" Type="Operation">Eq</TemplateValue>
            <Parameter Name="iN1">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_12" />
                    </Symbol>
                </Access>
            </Parameter>
            ...
        </Instruction>
    </Access>
</StlStatement>
```

Multiinstances used in calls

As of V14 SP1 the import is aborted if the multiinstance used in a call does not exists.

The following code shows an xml example where the multiinstance is defined correctly in the interface section:

```
<SW.Blocks.FB ID="0">
    <AttributeList>
        <Interface>
            <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
                <Section Name="Input" />
                <Section Name="Output" />
                <Section Name="InOut" />
                <Section Name="Static">
                    <!-- The next line must be present if multiinstance is used in code-->
                    <Member Name="Static_1" Datatype="&quot;Block_2&quot;" />
                </Section>
    ....
    <StlStatement UId="22">
        <StlToken Text="CALL" />
        <Access Scope="Call">
            <CallInfo BlockType="FB">
                <!-- Multiinstace usage-->
                <Instance Scope="LocalVariable">
                    <Component Name="Static_1" />
                </Instance>
                <Parameter Name="Input_1" Section="Input" Type="Int">
                    <Access Scope="GlobalVariable">
                        <Symbol>
                            <Component Name="Tag_9" />
                        </Symbol>
                    </Access>
                </Parameter>
            </CallInfo>
        </Access>
    </StlStatement>
```

Template cardinalities in STL

In STL the template cardinalities for every instruction has a fixed default value which is the only valid value. As of V14 SP1 the import is aborted if another value is used for the cardinality.

Importing indirect access

As of V14 SP1 indirect access can only be imported where they can be compiled.

```
<StlStatement UId="22">
    <StlToken Text="L" />
    <Access Scope="Address">
        <Indirect Width="Word" Area="Memory">
            <Access Scope="LocalVariable">
                <Symbol>
                    <Component Name="Temp_1" />
                </Symbol>
            </Access>
        </Indirect>
    </Access>
</StlStatement>
```

Importing statuswords

As of V14 SP1 the statusword can only be imported at statements where they are supported.

- L - Supported statusword: STW
- T - Supported statusword: STW
- A - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- AN - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- O - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- ON - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- X - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- XN - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU

Note

Most statuswords are only useful on 300 and 400 plcs.

Empty statements

The import is aborted if a statement does not have a node <StlStatement/>. In case of an empty statement, add the <StlToken Text="Empty_Line" /> node.

The import is aborted if an empty statement has comments. For a statement with only comments use the <StlToken Text="COMMENT" />.

```
<!-- Declaration of an empty statement -->
<StlStatement UId="23">
    <StlToken Text="EMPTY_LINE" />
</StlStatement>

<!-- Declaration of a statement with only comments-->
<StlStatement UId="22">
    <LineComment>
        <Text>Comment number 1</Text>
    </LineComment>
    <StlToken Text="COMMENT" />
</StlStatement>
```

7.6.4.6 Block attribute changes

Changes in general attributes

AutoNumber has got a new default value (false) at classic OBs

HeaderVersion has got a new type System.Version (instead of String)

IsKnowHowProtected is applied for user defined data types as well

ILibraryTypeInstance.ConnectedVersion, ILibraryTypeInstance.Dependencies, ILibraryTypeInstance.Dependents are eliminated from the table of general attributes because they are neither exported in XML nor accessible via API.

MemoryLayout gets new default: Standard in classic PLCs and Optimized on plus PLCs

Number is applied for user defined data types and it is represented in XML and accessible via API as well

Changes in specific attributes

IsOnlyStoredInLoadMemory and IsWriteProtectedInAS became read-only for IDBofUDT if it belongs to a system library element.

OfSystemLibElement and OfSystemLibVersion are relocated from general to specific attributes

OfSystemLibVersion has got a new type System.Version (instead of String)

ParameterPassing remains read-write at FCs and FBs only if

- ProgrammingLanguage is STL and
- MemoryLayout is standard and
- interface is empty

GraphVersion has got a new type System.Version (instead of String)

a new attribute called ExtensionBlockName is introduced for FBs written in Graph as of Graph version V4

a new attribute called InitialValuesAcquisition is introduced for FBs written in Graph as of Graph version V4

a new attribute called IsWriteProtected is introduced for code blocks

DownloadWithoutReinit became read-only and also applied for IDBofFBs

Supervisions became read-only on IDBofFBs .

Changes in enums

The enum values for ProgrammingLanguage are changed as follows:

- a new enum value F_CALL is introduced
- a new enum value Motion_DB is introduced for Motion technological object
- GRAPH_SEQUENCE, GRAPH_ACTIONS, GRAPH_ADDINFOS are deleted from the enum. They are replaced with GRAPH.

The enum values for BlockType are changed as follows:

- the values OB, FC, DB, SFC are deleted because this enum is only used at InstanceOfType attribute

7.7 Major changes in V14

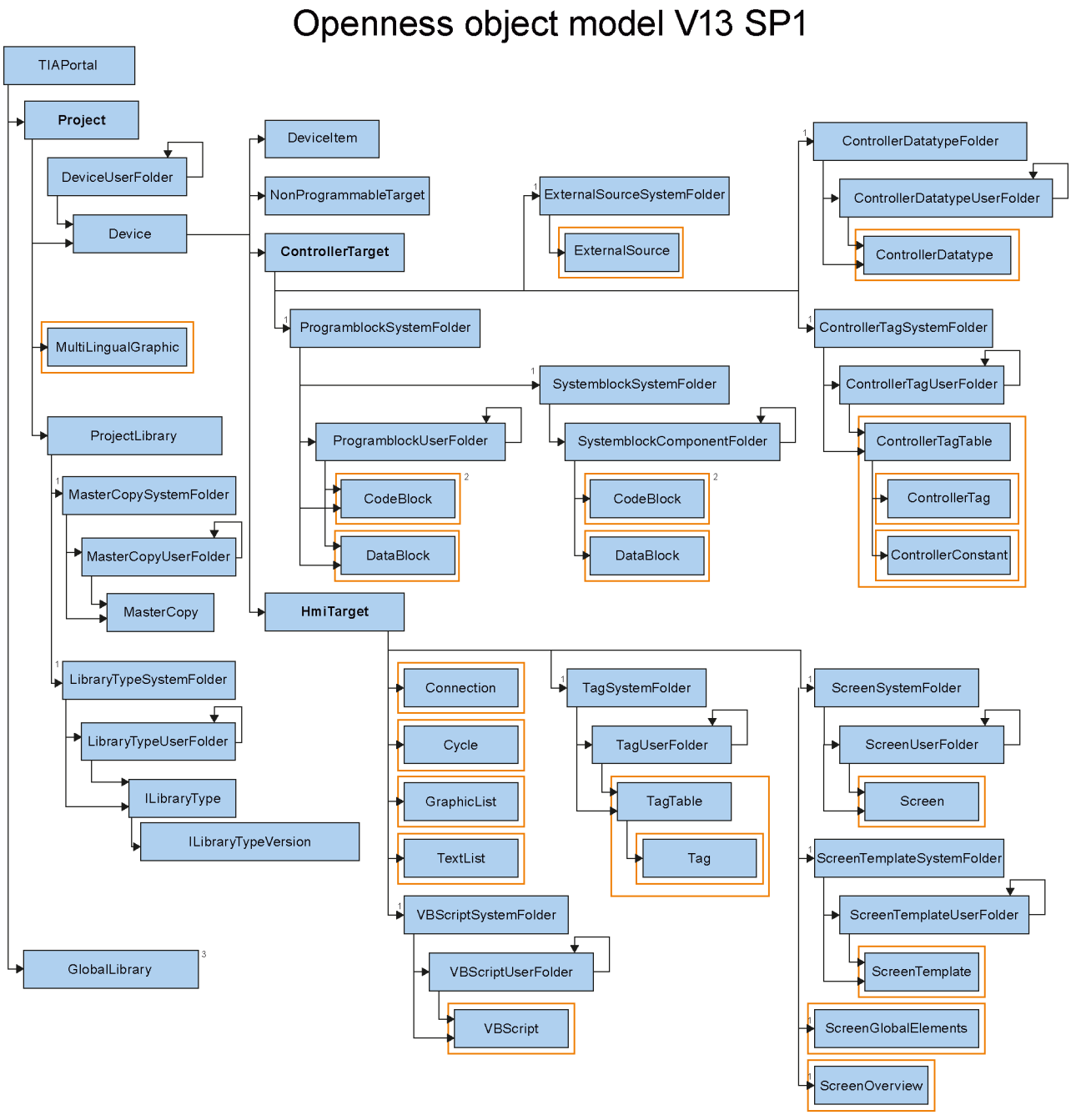
7.7.1 Major changes of the object model

Object model of TIA Portal Openness V13 SP1 and older

In order to allow you a comparison between the old and the new object model of TIA Portal Openness, the diagram below describes the object model of TIA Portal V13 SP1.

Note

The object model described on the diagram is obsolete, for information about the object model of TIA Portal Openness V14 SP1 refer to TIA Portal Openness object model (Page 53)


Openness object model V13 SP1
TIAPortal
Project
DeviceUserFolder
Device
DeviceItem
NonProgrammableTarget
ControllerTarget
ExternalSourceSystemFolder
ExternalSource
ControllerDatatypeFolder
ControllerDatatypeUserFolder
ControllerDatatype
MultiLingualGraphic
ProgramblockSystemFolder
ControllerTagSystemFolder
ControllerTagUserFolder
ControllerTagTable
ControllerTag
ControllerConstant
SystemblockSystemFolder
ProgramblockUserFolder
SystemblockComponentFolder
CodeBlock
DataBlock
ProjectLibrary
MasterCopySystemFolder
MasterCopyUserFolder
MasterCopy
HmiTarget
LibraryTypeSystemFolder
LibraryTypeUserFolder
ILibraryType
ILibraryTypeVersion
Connection
Cycle
GraphicList
TextList
VBScriptSystemFolder
VBScriptUserFolder
VBScript
TagSystemFolder
TagUserFolder
TagTable
Tag
ScreenSystemFolder
ScreenUserFolder
Screen
ScreenTemplateSystemFolder
ScreenTemplateUserFolder
ScreenTemplate
ScreenGlobalElements
ScreenOverview
GlobalLibrary

7.7.2 Before updating an application to TIA Portal Openness V14

Application

Before updating an application to TIA Portal Openness V14 change the following settings:

1. Adapt thr references to the V14 API by adding the following TIA Portal Openness APIs:
   - `Siemens.Engineering`
   - `Siemens.Engfineering.Hmi`
2. Change the .Net framework of your Visual Studio to version 4.6.1
3. Update the assembly resolve method by adapting the new installation path of the TIA Portal.
   - If you have evaluated from the registry, adapt the new key, according to the following example:
     `"HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\_InstalledSW\TIAP14\TIA_Opns\..."`
   - If you are using the application configuration file, adapt the paths to the new installation path.

7.7.3 Major string changes

Introduction

The following changes were made in TIA Portal Openness V14, which may impact your existing applications:

Change	Required program code adjustment

Compile methods have been changed.	Change the compile methods according to the following example:
	• TIA Portal Openness V13 SP1(obsolete):
	`controllerTarget.Compile(CompilerOptions.Software, BuildOptions.Rebuild);`
	• TIA Portal Openness V14:
	`plcSoftware.GetService<ICompilable>().Compile();`

New namespaces have been added.	1. Add the following namespace statements:
	`Siemens.Engineering.SW.Blocks;`
	`Siemens.Engineering.SW.ExternalSources;`
	`Siemens.Engineering.SW.Tags;`
	`Siemens.Engineering.SW.Types;`
	2. Remove the `using ControllerTarget = Siemens.Engineering.HW.ControllerTarget` namespace statement.
	3. Compile the application.

Change	Required program code adjustment

`ControllerTarget` has been replaced by `PlcSoftware` and functionality has been changed in some cases.	1. Review the code examples in the documentation which belong to your application functionality.
	2. Update the program code of your TIA Portal Openness application according to the following example:
	– TIA Portal Openness V13 SP1(obsolete):
	`ControllerTarget controllerTarget = deviceItem as ControllerTarget`
	– TIA Portal Openness V14:
	`PlcSoftware plcSoftware = deviceItem.GetService<SoftwareContainer>().Software as PlcSoftware`
	3. Compile the application.

Objects have been replaced.	1. Search and replace the following objects:
	`DeviceUserFolderAggregation = DeviceUserGroupComposition`
	`DeviceFolders = DeviceGroups`
	`DeviceUserFolder = DeviceUserGroup`
	`ProgramblockSystemFolder = PlcBlockSystemGroup`
	`ProgramblockUserFolder = PlcBlockUserGroup`
	`IBlock = PlcBlock`
	`ControllerDatatypeSystemFolder = PlcTypeSystemGroup`
	`ControllerDatatypeUserFolder = PlcTypeUserGroup`
	`ControllerDatatype = PlcType`
	`ControllerTagSystemFolder = PlcTagTableSystemGroup`
	`ControllerTagUserFolder = PlcTagTableUserGroup`
	`ControllerTagTable = PlcTagTable`
	`ControllerTag = PlcTag`
	`ControllerConstant = PlcConstant`
	`ExternalSourceSystemFolder = PlcExternalSourceSystemGroup`
	`ExternalSource = PlcExternalSource`
	`IOnline = OnlineProvider`
	`ILibraryType = LibraryType`
	2. Compile the application.

Change	Required program code adjustment

Aggregations have been replaced by compositions.	1. Replace every `Aggregation` of your code by `Composition` according to the following examples:
	`ProjectAggregation = ProjectComposition`
	`IDeviceAggregation = IDeviceComposition`
	`TagTableAggregation = TagTableComposition`
	`CycleAggregation = CycleComposition`
	`GraphicListAggregation = GraphicListComposition`
	`TextListAggregation = TextListComposition`
	`ConnectionAggregation = ConnectionComposition`
	`MultiLingualGraphicAggregation = MultiLingualGraphicComposition`
	`UpdateCheckResultMessageAggregation = UpdateCheckResultMessageComposition`
	2. Compile the application.

Folders have been replaced by groups in every relationship except HMI devices.	1. Replace every `Folder` in your program code by `Group` except code parts which concern to HMI devices. 2. Compile the application.
The `GetAttributeNames` method has been replaced by the `GetAttributeInfos` method.	1. Use `IList<EngineeringAttributeInfo> IEngineeringObject.GetAttributeInfos(AttributeAccessMode attributeAccessMode);` to determinate attributes. 2. Compile the application. For more detailed information, refer to Determining the object structure and attributes (Page 145).
The `Close` method for closing an object has changed.	1. Replace `project.Close(CloseMode.PromptIfModified);` by `project.Close();`. 2. Compile the application. For more detailed information, refer to Closing a project (Page 158).

Change	Required program code adjustment

Simultaneous access has been replaced by exclusive access and transactions.	1. Replace simultaneous access by exclusive access and transactions according to the following examples:
	– TIA Portal Openness V13 SP1(obsolete):
	`tiaProject.StartTransaction("Reseting project to default");`
	`...`
	`tiaProject.CommitTransaction();`
	– TIA Portal Openness V14:
	`//Use exclusive access to avoid user changes`
	`ExclusiveAccess exclusiveAccess = tiaPortal.ExclusiveAccess();`
	`...`
	`exclusiveAccess.Dispose();`
	`//Use transaction to be able to rollbank changes:`
	`Transaction transaction = exclusiveAccess.Transaction(tiaProject, "Compiling device");`
	`transaction.CommitOnDispose();`
	2. Compile the application.
	See Exclusive access (Page 104) and Transaction handling (Page 115) for further information.

Online access to the CPU has been changed	1. Change the online access to the CPU according to the following examples:
	– TIA Portal Openness V13 SP1(obsolete):
	`((IOnline)controllerTarget).GoOffline();`
	– TIA Portal Openness V14:
	`((DeviceItem) plcSoftware.Parent.Parent).GetService<Onlin eProvider>().GoOffline();`
	2. Compile the application.

The hardware configuration has been changed	1. Change the hardware configuration: `Device.Elements = Device.Items` 2. Remove the following hardware attributes: – `Device.InternalDeviceItem` – `Device.SubType` 3. Compile the application.

See also

Handling exceptions (Page 1187)

Connecting to the TIA Portal (Page 82)

7.7.4 Import of files generated with TIA Portal Openness V13 SP1 and previous

Application

When you try to import files which were generated with TIA Portal Openness V13 SP1 or previous an exeption will be thrown because of incompatibility. This is caused by changes on HMI tags and HMI screen items. The following tables are showing the main attribute changes, for more detailed information refer to the chapter "Creating screens Working with objects and object groups > Working with objects > Configuring ranges" of the TIA Portal online help:

Changes of HMI tags

The following table shows the main changes of HMI tag attributes:

Removed attributes	Added attributes
RangeMaximumType	LimitUpper2Type.
RangeMaximum	LimitUpper2.
RangeMinimumType	LimitLower2Type.
RangeMinimum	LimitLower2.
	LimitUpper1Type
	LimitUpper1
	LimitLower1Type
	LimitLower1

Changes of HMI screen items

The following table shows the main changes of slider attributes:

Removed attributes	Added attributes
	RangeLower1Color
	RangeLower1Enabled
	RangeLower2Color
	RangeLower2Enabled
	RangeNormalColor
	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper2Color
	RangeUpper2Enabled
	ScalePosition
	ShowLimitLines
	ShowLimitMarkers
	ShowLimitRanges

The following table shows the main changes of gauge attributes:

Removed attributes	Added attributes
DangerRangeColor	RangeLower1Color
DangerRangeStart	RangeLower1Enabled
DangerRangeVisible	RangeLower2Color
WarningRangeColor	RangeLower2Enabled
WarningRangeStart	RangeNormalColor
WarningRangeVisible	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper1Start
	RangeUpper2Color
	RangeUpper2Enabled
	RangeUpper2Start

The following table shows the main changes of bar attributes:

Removed attributes	Added attributes
AlarmLowerLimitColor	RangeLower1Color
AlarmUpperLimitColor	RangeLower1Enabled
	RangeLower2Color
	RangeLower2Enabled
	RangeNormalColor
	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper2Color
	RangeUpper2Enabled

Index

E

F

G

H

I

L

M

O

P

W

X