

# SIEMENS

## SIMATIC

### Openness: Automating creation of projects

System Manual

<u>Security note</u>	<b>1</b>
<u>Readme TIA Portal Openness</u>	<b>2</b>
<u>What's new in TIA Portal Openness?</u>	<b>3</b>
<u>Basics</u>	<b>4</b>
<u>Introduction</u>	<b>5</b>
<u>Configurations</u>	<b>6</b>
<u>TIA Portal Openness API</u>	<b>7</b>
<u>Export/import</u>	<b>8</b>
<u>Major Changes</u>	<b>9</b>

Printing the Online Help

## 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.

<b>⚠ DANGER</b>
indicates that death or severe personal injury <b>will</b> result if proper precautions are not taken.

<b>⚠ WARNING</b>
indicates that death or severe personal injury <b>may</b> result if proper precautions are not taken.

<b>⚠ CAUTION</b>
indicates that minor personal injury can result if proper precautions are not taken.

<b>NOTICE</b>
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:

<b>⚠ WARNING</b>
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 AG. 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.

# Table of contents

<b>1</b>	<b>Security note</b>	<b>11</b>
<b>2</b>	<b>Readme TIA Portal Openness</b>	<b>13</b>
2.1	Readme	13
2.2	Major changes in TIA Portal Openness V15.1	16
2.3	Announcement of major changes in future releases	19
2.4	Hints for writing long-term stable code	20
<b>3</b>	<b>What's new in TIA Portal Openness?</b>	<b>23</b>
<b>4</b>	<b>Basics</b>	<b>25</b>
4.1	Requirements for TIA Portal Openness	25
4.2	Installation	27
4.2.1	Installing TIA Openness	27
4.2.2	Adding users to the "Siemens TIA Openness" user group	28
4.2.3	Accessing the TIA Portal	33
4.3	Openness tasks	34
4.3.1	Applications	34
4.3.2	Export/import	35
4.4	Object list	36
4.5	Standard libraries	40
4.6	Notes on performance of TIA Portal Openness	41
<b>5</b>	<b>Introduction</b>	<b>43</b>
<b>6</b>	<b>Configurations</b>	<b>45</b>
<b>7</b>	<b>TIA Portal Openness API</b>	<b>49</b>
7.1	Introduction	49
7.2	Programming steps	50
7.3	TIA Portal Openness object model	51
7.4	Blocks and types of the TIA Portal Openness object model	56
7.5	Hierarchy of hardware objects of the object model	64
7.6	Information about installed TIA Portal Openness versions	66
7.7	Example program	67
7.8	Use of the code examples	72
7.9	General functions	74
7.9.1	TIA Portal Openness IntelliSense support	74
7.9.2	Connecting to the TIA Portal	74
7.9.3	TIA Portal Openness firewall	79

7.9.4	Event handlers.....	80
7.9.5	Program-controlled acknowledgement of dialogs with system events.....	82
7.9.6	Terminating the connection to the TIA Portal.....	83
7.9.7	Diagnostic interfaces on TIA Portal.....	84
7.9.8	Exclusive access.....	89
7.9.9	Transaction handling.....	91
7.9.10	Creating a DirectoryInfo/FileInfo object.....	94
7.9.11	Self-description support for attributes, navigators, actions, and services.....	94
7.10	Functions for projects and project data.....	97
7.10.1	Opening a project.....	97
7.10.2	Creating a project.....	101
7.10.3	Accessing general settings of the TIA Portal.....	102
7.10.4	Accessing read-only TIA Portal project.....	106
7.10.5	Accessing languages.....	107
7.10.6	Determining the object structure and attributes.....	109
7.10.7	Access software target .....	111
7.10.8	Accessing and enumerating multilingual texts.....	112
7.10.9	Read project related attributes.....	113
7.10.10	Deleting project graphics.....	116
7.10.11	Compiling a project.....	116
7.10.12	Saving a project.....	119
7.10.13	Closing a project.....	120
7.11	Functions for Connections.....	122
7.11.1	Configurable attributes of a port-to-port connection.....	122
7.12	Functions on libraries.....	125
7.12.1	Functions for objects and instances.....	125
7.12.2	Accessing global libraries.....	126
7.12.3	Accessing global library languages.....	128
7.12.4	Opening libraries.....	130
7.12.5	Enumerating open libraries.....	131
7.12.6	Saving and closing libraries.....	132
7.12.7	Archiving and retrieving a library.....	133
7.12.8	Creating global libraries.....	136
7.12.9	Accessing folders in a library.....	137
7.12.10	Accessing types.....	140
7.12.11	Accessing type versions.....	142
7.12.12	Accessing instances.....	146
7.12.13	Accessing master copies.....	148
7.12.14	Create master copy from a project in library.....	151
7.12.15	Create an object from a master copy.....	152
7.12.16	Copying master copies.....	154
7.12.17	Determining out-of-date type instances.....	155
7.12.18	Updating the project.....	158
7.12.19	Updating a library.....	160
7.12.20	Deleting library content.....	161
7.13	Functions for accessing devices, networks and connections.....	164
7.13.1	Open the "Devices & networks" editor.....	164
7.13.2	Querying PLC and HMI targets.....	165
7.13.3	Accessing attributes of an address object.....	166
7.13.4	Accessing the channels of a module.....	169

---

7.13.5	Working with associations.....	170
7.13.6	Working with compositions.....	171
7.13.7	Verifying object equality.....	172
7.13.8	Read operations for attributes.....	173
7.14	Functions on networks.....	175
7.14.1	Creating a subnet.....	175
7.14.2	Accessing subnets.....	176
7.14.3	Accessing internal subnets.....	177
7.14.4	Get type identifier of subnets.....	178
7.14.5	Accessing attributes of a subnet.....	179
7.14.6	Deleting a global subnet.....	185
7.14.7	Enumerate all participants of a subnet.....	185
7.14.8	Enumerate io systems of a subnet.....	186
7.14.9	Accessing nodes.....	186
7.14.10	Accessing attributes of a node.....	187
7.14.11	Connecting a node to a subnet.....	191
7.14.12	Disconnect a node from a subnet.....	191
7.14.13	Creating an io system.....	192
7.14.14	Accessing the attributes of an io system.....	193
7.14.15	Connecting an io connector to an io system.....	193
7.14.16	Get master system or io system of an interface.....	194
7.14.17	Get an IO Controller.....	195
7.14.18	Get an IO Connector.....	196
7.14.19	Disconnecting an io connector from an io system or a dp mastersystem.....	196
7.14.20	Accessing attributes of a dp mastersystem.....	197
7.14.21	Accessing attributes of a profinet io system.....	198
7.14.22	Deleting a dp mastersystem.....	199
7.14.23	Deleting a profinet io system.....	200
7.14.24	Creating a dp master system.....	200
7.14.25	Accessing port interconnection information of port device item.....	201
7.14.26	Attributes of port inter-connection.....	202
7.14.27	Accessing the attributes of a port.....	205
7.14.28	Enumerate dp master systems of a subnet.....	206
7.14.29	Enumerate assigned io connectors.....	207
7.14.30	Connecting a dp io connector to a dp mastersystem.....	208
7.15	Functions on devices.....	209
7.15.1	Mandatory attributes of devices.....	209
7.15.2	Get type identifier of devices and device items.....	210
7.15.3	Creating a device.....	213
7.15.4	Enumerating devices.....	214
7.15.5	Accessing devices.....	217
7.15.6	Deleting a device.....	219
7.16	Functions on device items.....	221
7.16.1	Mandatory attributes of device items.....	221
7.16.2	Creating and plugging a device item.....	222
7.16.3	Moving device items into another slot.....	226
7.16.4	Copying a device item.....	227
7.16.5	Deleting a device item.....	228
7.16.6	Enumerate device items.....	228
7.16.7	Accessing device items.....	230
7.16.8	Accessing device item as interface.....	234

7.16.9	Accessing attributes of an I/O device interface.....	235
7.16.10	Accessing attributes of IoController.....	237
7.16.11	Accessing attributes of IoConnector.....	238
7.16.12	Accessing address controller.....	240
7.16.13	Accessing addresses.....	241
7.16.14	Accessing hardware identifiers.....	243
7.16.15	Accessing hardware identifier controller.....	244
7.16.16	Accessing channels of device items.....	245
7.17	Functions for accessing the data of an HMI device.....	247
7.17.1	Screens.....	247
7.17.1.1	Creating user-defined screen folders.....	247
7.17.1.2	Deleting a screen from a folder.....	247
7.17.1.3	Deleting a screen template from a folder.....	248
7.17.1.4	Deleting all screens from a folder.....	249
7.17.2	Cycles.....	250
7.17.2.1	Deleting a cycle.....	250
7.17.3	Text lists.....	250
7.17.3.1	Deleting a text list.....	250
7.17.4	Graphic lists.....	251
7.17.4.1	Deleting a graphic list.....	251
7.17.5	Connections.....	252
7.17.5.1	Deleting a connection.....	252
7.17.6	Tag table.....	252
7.17.6.1	Creating user-defined folders for HMI tags.....	252
7.17.6.2	Enumerating tags of an HMI tag table.....	253
7.17.6.3	Deleting an individual tag from an HMI tag table.....	253
7.17.6.4	Deleting a tag table from a folder.....	254
7.17.7	VB scripts.....	255
7.17.7.1	Creating user-defined script folders.....	255
7.17.7.2	Deleting a VB script from a folder.....	255
7.17.8	Deleting a user-defined folder of an HMI device .....	256
7.18	Functions for accessing the data of a PLC device.....	257
7.18.1	Determining the status of a PLC.....	257
7.18.2	Accessing parameters of an online connection.....	258
7.18.3	Setting PLC online of R/H system.....	262
7.18.4	Accessing software container from primary PLC of R/H system .....	264
7.18.5	Downloading PLCs of R/H System.....	265
7.18.6	Functions for downloading data to PLC device.....	271
7.18.6.1	Downloading hardware and software components to PLC device.....	271
7.18.6.2	Running and stopping PLC.....	281
7.18.6.3	Supporting callbacks.....	282
7.18.6.4	Protecting PLC through password.....	284
7.18.6.5	Handling PLC block binding passwords.....	285
7.18.7	Uploading hardware, software and files to PLC device.....	286
7.18.8	Accessing fingerprints.....	292
7.18.9	Comparing PLC software.....	293
7.18.10	Comparing PLC hardware.....	296
7.18.11	Establishing or disconnecting the online connection to the PLC.....	297
7.18.12	Blocks.....	299
7.18.12.1	Querying the "Program blocks" group.....	299
7.18.12.2	Querying the system group for system blocks.....	299

7.18.12.3	Enumerating system subgroups.....	300
7.18.12.4	Enumerating user-defined block groups.....	301
7.18.12.5	Enumerating all blocks.....	302
7.18.12.6	Querying information of a block/user data type.....	303
7.18.12.7	Setting and removing protections from a block.....	305
7.18.12.8	Deleting block.....	307
7.18.12.9	Creating group for blocks.....	308
7.18.12.10	Deleting group for blocks.....	309
7.18.12.11	Accessing attributes of all blocks.....	309
7.18.12.12	Creating a ProDiag-FB.....	310
7.18.12.13	Accessing supervisions and properties of ProDiag-FB.....	311
7.18.12.14	Reading ProDiag-FB blocks and attributes.....	313
7.18.12.15	Adding an external file.....	313
7.18.12.16	Generate source from block.....	314
7.18.12.17	Generating blocks from source.....	316
7.18.12.18	Deleting user data type.....	317
7.18.12.19	Deleting an external file.....	318
7.18.12.20	Starting the block editor.....	319
7.18.13	Technology objects.....	319
7.18.13.1	Overview of functions for technology objects.....	319
7.18.13.2	Overview of technology objects and versions.....	320
7.18.13.3	Overview of data types.....	322
7.18.13.4	Querying the composition of technology objects.....	323
7.18.13.5	Creating technology object.....	323
7.18.13.6	Deleting technology object.....	324
7.18.13.7	Compiling technology object.....	325
7.18.13.8	Enumerating technology object.....	326
7.18.13.9	Finding technology object.....	327
7.18.13.10	Enumerating parameters of technology object.....	328
7.18.13.11	Finding parameters of technology object.....	328
7.18.13.12	Reading parameters of technology object.....	329
7.18.13.13	Writing parameters of technology object.....	330
7.18.13.14	S7-1200 Motion Control.....	331
7.18.13.15	S7-1500 Motion Control.....	339
7.18.13.16	PID control.....	357
7.18.13.17	Counting.....	358
7.18.13.18	Easy Motion Control.....	358
7.18.14	Tags and Tag tables.....	359
7.18.14.1	Starting the "PLC Tags" editor.....	359
7.18.14.2	Querying system groups for PLC tags.....	360
7.18.14.3	Creating PLC tag table.....	360
7.18.14.4	Enumerating user-defined groups for PLC tags.....	361
7.18.14.5	Creating user-defined groups for PLC tags.....	362
7.18.14.6	Deleting user-defined groups for PLC tags.....	363
7.18.14.7	Enumerating PLC tag tables in a folder.....	363
7.18.14.8	Querying information from a PLC tag table.....	364
7.18.14.9	Reading the time of the last changes of a PLC tag table.....	366
7.18.14.10	Deleting a PLC tag table from a group.....	366
7.18.14.11	Enumerating PLC tags.....	367
7.18.14.12	Accessing PLC tags.....	367
7.18.14.13	Accessing PLC constants.....	369
7.19	Functions on OPC.....	372

7.19.1	Configuring OPC UA server secure communication protocol.....	372
7.19.2	Setting OPC UA security policy.....	374
7.20	SiVArc Openness.....	376
7.20.1	Introduction.....	376
7.21	Openness for CP 1604/CP 1616/CP 1626.....	377
7.22	Openness for SIMATIC Ident.....	381
7.22.1	Openness for SIMATIC Ident.....	381
7.22.2	ASM 456.....	382
7.22.3	ASM 475.....	387
7.22.4	RF120C.....	388
7.22.5	RF170C.....	396
7.22.6	RF180C.....	401
7.22.7	RF18xC.....	403
7.22.8	RF615R/RF680R/RF685R.....	405
7.22.9	MV400/MV500.....	406
7.23	Exceptions.....	407
7.23.1	Handling exceptions.....	407
<b>8</b>	<b>Export/import.....</b>	<b>411</b>
8.1	Overview.....	411
8.1.1	Basic principles of importing/exporting.....	411
8.1.2	Field of application for Import/Export.....	413
8.1.3	Version Specific Simatic ML Import.....	414
8.1.4	Editing the XML file.....	415
8.1.5	Exporting configuration data.....	415
8.1.6	Importing configuration data.....	417
8.2	Import/export of project data.....	419
8.2.1	Project graphics.....	419
8.2.1.1	Exporting/importing graphics.....	419
8.2.1.2	Exporting all graphics of a project.....	420
8.2.1.3	Importing graphics to a project.....	421
8.2.2	Project texts.....	422
8.2.2.1	Export of project texts.....	422
8.2.2.2	Import of project texts.....	423
8.3	Importing/exporting data of an HMI device.....	425
8.3.1	Structure of an XML file.....	425
8.3.2	Structure of the data for importing/exporting.....	427
8.3.3	Cycles.....	430
8.3.3.1	Exporting cycles.....	430
8.3.3.2	Importing cycles.....	431
8.3.4	Tag tables.....	432
8.3.4.1	Exporting HMI tag tables.....	432
8.3.4.2	Importing HMI tag table.....	435
8.3.4.3	Exporting an individual tag from an HMI tag table.....	436
8.3.4.4	Importing an individual tag into an HMI tag table.....	437
8.3.4.5	Special considerations for the export/import of HMI tags.....	437
8.3.5	VB scripts.....	439
8.3.5.1	Exporting VB scripts.....	439
8.3.5.2	Exporting VB scripts from a folder.....	440



8.3.5.3	Importing VB scripts.....	441
8.3.6	Text lists.....	442
8.3.6.1	Exporting text lists from an HMI device.....	442
8.3.6.2	Importing a text list into an HMI device.....	443
8.3.6.3	Advanced XML formats for export/import of text lists.....	444
8.3.7	Graphic lists.....	446
8.3.7.1	Exporting graphic lists.....	446
8.3.7.2	Importing a graphic list.....	446
8.3.8	Connections.....	447
8.3.8.1	Exporting connections.....	447
8.3.8.2	Importing connections.....	448
8.3.9	Screens.....	449
8.3.9.1	Overview of exportable screen objects.....	449
8.3.9.2	Exporting all screens of an HMI device.....	453
8.3.9.3	Exporting a screen from a screen folder.....	454
8.3.9.4	Importing screens to an HMI device.....	456
8.3.9.5	Exporting permanent areas.....	458
8.3.9.6	Importing permanent areas.....	459
8.3.9.7	Exporting all screen templates of an HMI device.....	460
8.3.9.8	Exporting screen templates from a folder.....	461
8.3.9.9	Importing screen templates.....	463
8.3.9.10	Exporting a pop-up screen.....	464
8.3.9.11	Importing a pop-up screen.....	465
8.3.9.12	Exporting a slide-in screen.....	467
8.3.9.13	Importing a slide-in screen.....	468
8.3.9.14	Exporting a screen with a faceplate instance.....	469
8.3.9.15	Importing a screen with a faceplate instance.....	471
8.4	Importing/exporting data of a PLC device.....	475
8.4.1	Blocks.....	475
8.4.1.1	XML structure of the block interface section .....	475
8.4.1.2	Changes of the object model and XML file format.....	485
8.4.1.3	Exporting blocks .....	486
8.4.1.4	Exporting DBs with snapshots.....	492
8.4.1.5	Exporting blocks with know-how protection.....	495
8.4.1.6	Export/Import of SCL blocks.....	495
8.4.1.7	Export/Import of structured types of SCL blocks.....	509
8.4.1.8	Export/Import of SCL call blocks.....	515
8.4.1.9	Exporting failsafe blocks.....	532
8.4.1.10	Exporting system blocks.....	532
8.4.1.11	Exporting GRAPH blocks with multi-language text.....	533
8.4.1.12	Importing block.....	534
8.4.1.13	Importing blocks/UDT with open reference.....	535
8.4.1.14	Importing blocks/UDT for structural change object.....	537
8.4.2	Tag tables.....	538
8.4.2.1	Exporting PLC tag tables.....	538
8.4.2.2	Importing PLC tag table.....	539
8.4.2.3	Exporting an individual tag or constant from a PLC tag table.....	540
8.4.2.4	Importing an individual tag or constant into a PLC tag table.....	541
8.4.3	Exporting user data type.....	542
8.4.4	Importing user data type.....	543
8.4.5	Export of data in OPC UA XML format.....	544

8.5	Importing/exporting hardware data.....	546
8.5.1	AML file format.....	546
8.5.2	Pruned AML.....	546
8.5.3	Overview of the objects and parameters of the CAx import/export.....	548
8.5.4	Structure of the CAx data for importing/exporting.....	550
8.5.5	AML type identifiers.....	555
8.5.6	Export of CAx data.....	558
8.5.7	Export/Import of sub modules.....	562
8.5.8	Import of CAx data.....	566
8.5.9	Exceptions during import and export of CAx data.....	568
8.5.10	Round trip exchange of devices and modules.....	569
8.5.11	Export/Import topology.....	572
8.5.12	Export of a device object.....	574
8.5.13	Import of a device object.....	576
8.5.14	Export/Import of device with set address.....	579
8.5.15	Export/Import of device with channels.....	582
8.5.16	Export of device item objects.....	584
8.5.17	Import of device item objects.....	587
8.5.18	Export/Import of GSD/GSDML based devices and device items.....	590
8.5.19	Export/Import of subnets.....	594
8.5.20	Export/Import of PLC tags.....	600
8.5.21	Export/Import of IO-systems.....	602
8.5.22	Export/Import of multilingual comments.....	604
8.5.23	AML attributes versus TIA Portal Openness attributes.....	606
<b>9</b>	<b>Major Changes.....</b>	<b>609</b>
9.1	Major changes in TIA Portal Openness V15.....	609
9.2	Major changes in V14 SP1.....	611
9.2.1	Major changes in V14 SP1.....	611
9.2.2	Major changes in the object model.....	614
9.2.3	Changes on pilot functionality.....	618
9.2.4	Changes for export and import.....	623
9.2.4.1	Changes for export and import.....	623
9.2.4.2	Changes in API.....	623
9.2.4.3	Schema extension.....	624
9.2.4.4	Schema changes.....	627
9.2.4.5	Behaviour changes.....	630
9.2.4.6	Block attribute changes.....	641
9.3	Major changes in V14.....	643
9.3.1	Major changes of the object model.....	643
9.3.2	Before updating an application to TIA Portal Openness V14.....	645
9.3.3	Major string changes.....	645
9.3.4	Import of files generated with TIA Portal Openness V13 SP1 and previous.....	649
	<b>Index.....</b>	<b>651</b>

# Security note

## Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines, equipment and/or networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit

<http://www.siemens.com/industrialsecurity> (<http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx>)

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under

<http://www.siemens.com/industrialsecurity> (<http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx>)



# Readme TIA Portal Openness

## 2.1 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 V15.1\PublicAPI\V15.1\HW Parameter Description \Openness\_hardware\_parameter\_description.pdf

### 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

#### Multiuser

TIA Portal Openness doesn't support administrative multiuser operations. Because of that it's not recommended to use TIA Portal Openness in Multiuser projects. Be aware that there are TIA Portal Openness actions that actually interfere with the multiuser workflow that is enforced by the GUI of the TIA portal. If you want to make modifications with TIA Portal Openness, export the multiuser project to a single user project before.

#### 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.

## 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 script 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.

## Copying S7-1500 Motion Control technology objects

A copy of TO\_CamTrack, TO\_OutputCam or TO\_MeasuringInput from the project library or global library to the project is not possible.

## 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 Openness

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

## 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

## 2.2 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.EthernetDevice.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 TiaPortal instance with an MTA.

If TiaPortal is created or attached within an STA thread, all Openness objects associated with that Portal instance should 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 independent 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 may not be changed via Openness.

### Attribute `TransmissionRateAndDuplex`

Some faulty enum values for attribute `TransmissionRateAndDuplex` have been corrected.

### 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 `ChannellInfo` interface

Up to TIA Portal Openness V15 the `ChannellInfo` interface the number of available channels hasn't been correct for some modules.

### Access to ProDiag FB attributes

The following attributes of a ProDiag FB can be accessed 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 Openness V15.1.

## **R/H systems**

Download for R/H devices is available at device.

Online Provider is not available for R/H devices.

For PLC2 of an R/H system SoftwareContainer will not be available.

## 2.3 Announcement of major changes in future releases

### Announcement of changes

The TIA Portal Openness API will be changed in a later version. There is no need to change the code of your application immediately, because applications based on former releases will run without any restriction. But for new applications it is recommended to use the new functionality and to plan the recoding of your application, as of V17 the following methods will not be supported any longer.

- Type of compositions

### Type of compositions

The following types will be changed to indicate the snapshot behaviour:

- AddressAssociation
- AddressComposition
- AddressControllerAssociation
- ChannelComposition
- DeviceItemAssociation
- DeviceItemComposition
- HwIdentifierAssociation
- HwIdentifierComposition
- HwIdentifierControllerAssociation
- IoConnectorComposition
- IoControllerComposition

### Simocode Openness

Please note that while all Truth Table parameters will be available for use in Openness in V15.1, their implementation will change in V16.

Instead of having to set each bit individually, it will be possible to set all outputs bits of a Truth Table in one quick array operation.

If you want to keep using your existing Openness scripts in V16, you may have to adapt your code to the new situation.

## 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.

### Registry path and appconfig file

Modifications are necessary to change registry path and appconfig 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 V15\PublicAPI\V14  
SP1\Siemens.Engineering.dll”

To write long-term stable code, the registry path should be configurable and the appconfig 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 V15\PublicAPI\V14  
SP1\Siemens.Engineering.dll”

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

### Path of AmiHost

Modifications are necessary to change the path of AmiHost, for instance:  
“C:\Program Files\Siemens\Automation\Portal V14\bin  
\Siemens.Automation.Cax.AmiHost.exe”  
has to be changed to “C:\Program Files\Siemens\Automation\Portal V15\bin  
\Siemens.Automation.Cax.AmiHost.exe”

To write long-term stable code, the path of AmiHost 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  
\*.ap15

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 compare, compile or download results

The hierarchy and/or the order of compare, compile or download results 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: `UploadResult`. 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?

## New TIA Portal Openness object model

The following new features and innovations are available in TIA Portal Openness V15.1. You can find additional details on the various topics in the individual sections of the product documentation.

- **Openness DLLs of V14 SP1, V15 and V15.1 in scope of delivery**  
Because the Openness DLLs of V14 SP1, V15 and V15.1 are included in the scope of delivery, applications based on V14 SP1 and V15 also run in V15.1 without modification. To make use of the functions of V15.1, you must integrate the DLL of V15.1 and recompile the application
- **Siemens.Engineering.dll files are available for V14 SP1, V15, and V15.1.**  
They can be found in the installation directory under "PublicAPI\[version]".  
For example, the V14 SP1 dll can be found as "C:\Program Files\Siemens\Automation\Portal V15\_1\PublicAPI\V14 SP1\Siemens.Engineering.dll".
- **Accessing projects**  
Multiple projects in a TIA Portal instance can be opened via Openness.  
Projects can be archived and restored via Openness.  
For UMAC-protected projects Openness enables read access to objects. The same restrictions apply to this type of access as for a user with the "Read only" authorization.
- **Online/offline comparison**  
The online and offline comparison of data is possible via Openness.
- **Accessing protection levels**  
A protection level can be set or removed for blocks.
- **Accessing fingerprints**  
The fingerprint can be queried for blocks and PLC data types (UDTs).
- **Accessing names**  
The name can be set for blocks, data blocks and UDTs.
- **Fault-tolerant import**  
In addition to the strict import that can still be used, a fault-tolerant import is now available. The import is still possible even if linked user data types or called blocks do not match, for example.
- **Export and import**  
Watch tables as well as force tables can be exported and imported. Snapshots of the current values can be exported as XML from an offline DB. This means different snapshots can be compared with the help of the XML files.
- **ET200SP**  
Read and write access is possible for most attributes of the ET200SP modules.

- R/H systems  
Download to the primary PLC and the backup PLC is possible for R/H systems.
- Fail-safe PLC  
The upload is now also possible indirectly via NAT router.  
Recipes, archives, user files and the passwords of the protection levels are also taken into account during the upload.

For changes to the object model see Major Changes in TIA Portal Openness V15.1 (Page 16) for further information.

## **See also**

TIA Portal Openness object model (Page 51)

Announcement of major changes in future releases (Page 19)

Major changes in TIA Portal Openness V15.1 (Page 16)

Major changes in TIA Portal Openness V15 (Page 609)

Major changes in V14 SP1 (Page 611)

Major changes in V14 (Page 643)



## Basics

### 4.1 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 Openness" is installed on the PC.  
See Installing TIA Openness (Page 27)

#### 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 2015 Update 1 or later with .Net 4.6.2

#### Necessary user knowledge

- Knowledge as a system engineer
- Advanced knowledge of Microsoft Visual Studio 2015 Update 1 or later with .Net 4.6.2
- Advanced knowledge of C# / VB.net and .Net
- 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".

---

4.1 Requirements for TIA Portal Openness

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 28)

## 4.2 Installation

### 4.2.1 Installing TIA Openness

#### Introduction

The "TIA Openness" is installed via TIA portal setup program by selecting the TIA 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 Openness is already installed, the current version will be installed side by side.

---

#### Procedure

To install the TIA Openness, ensure the TIA 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 Openness checkbox.
3. Click "Next" and select the required option.

Follow the installation procedure of TIA portal to complete the TIA 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 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 28)).

---

## 4.2.2 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.

---

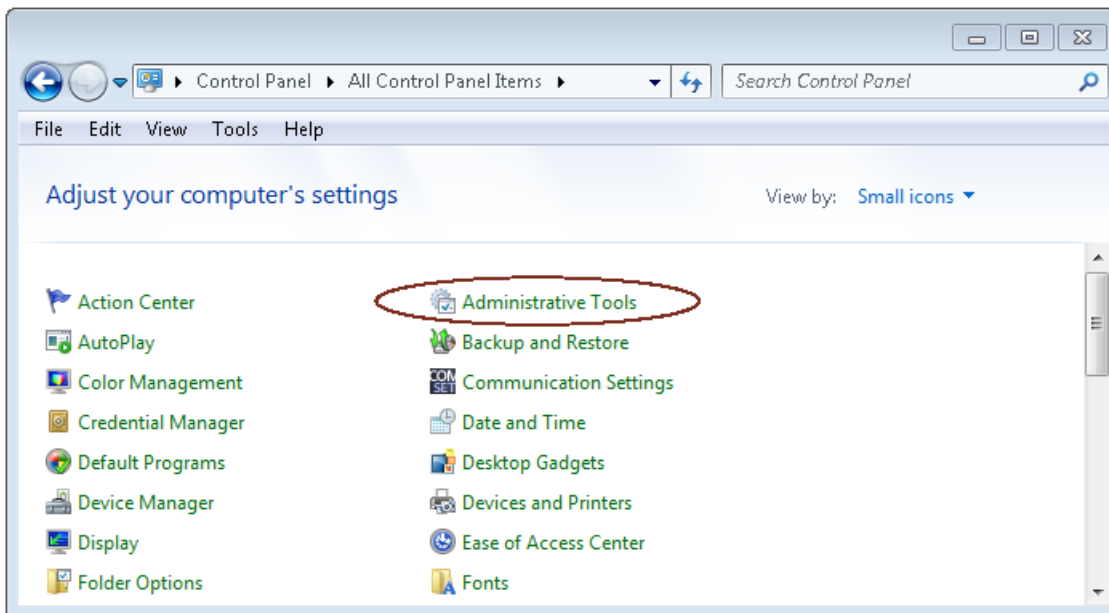
### Note

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

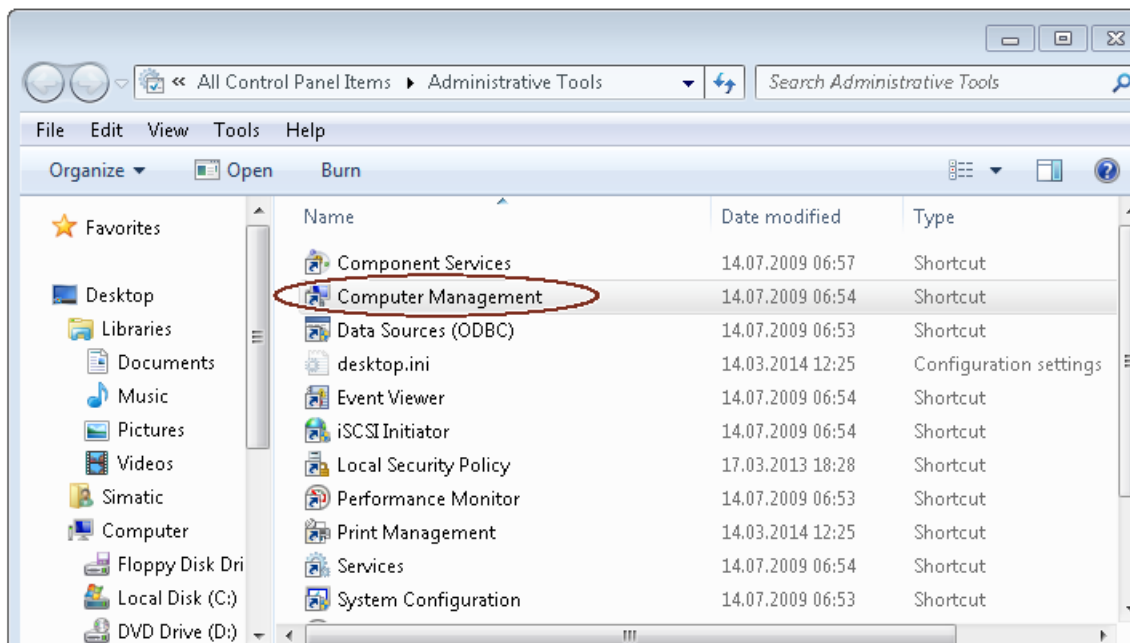
---

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.

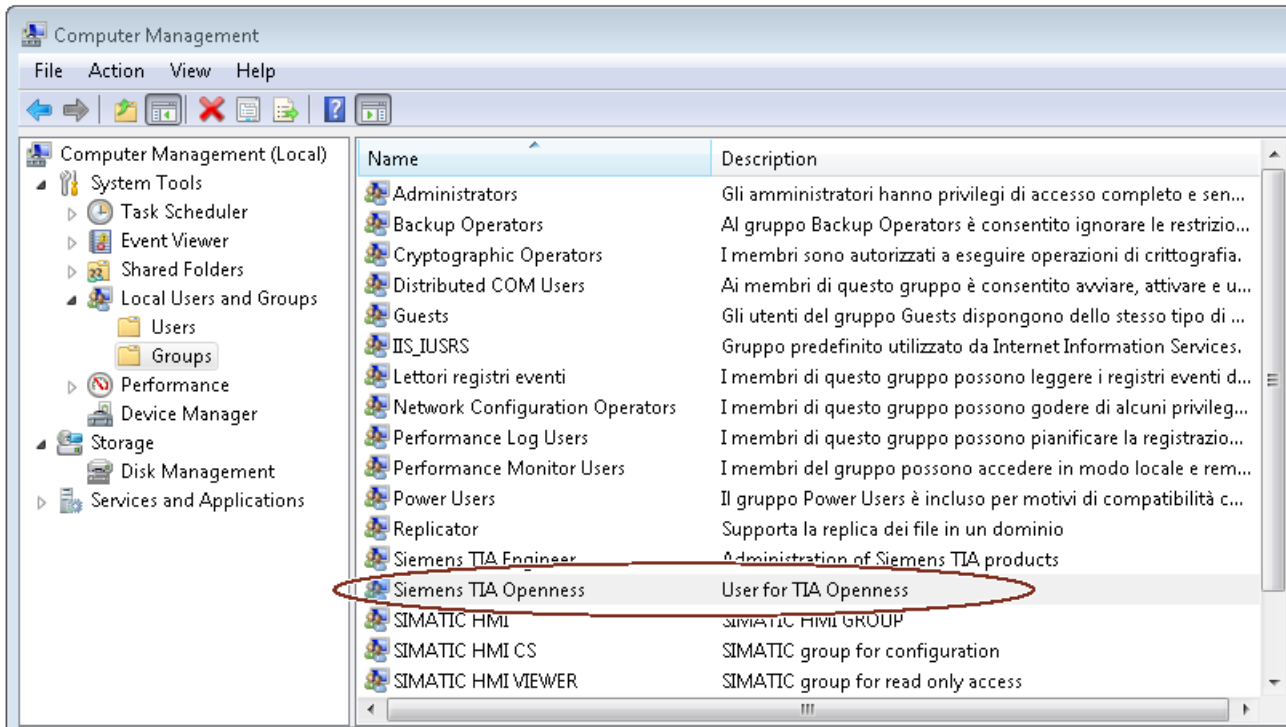


3. Click "Computer Management" to open the configuration dialog of the same name.

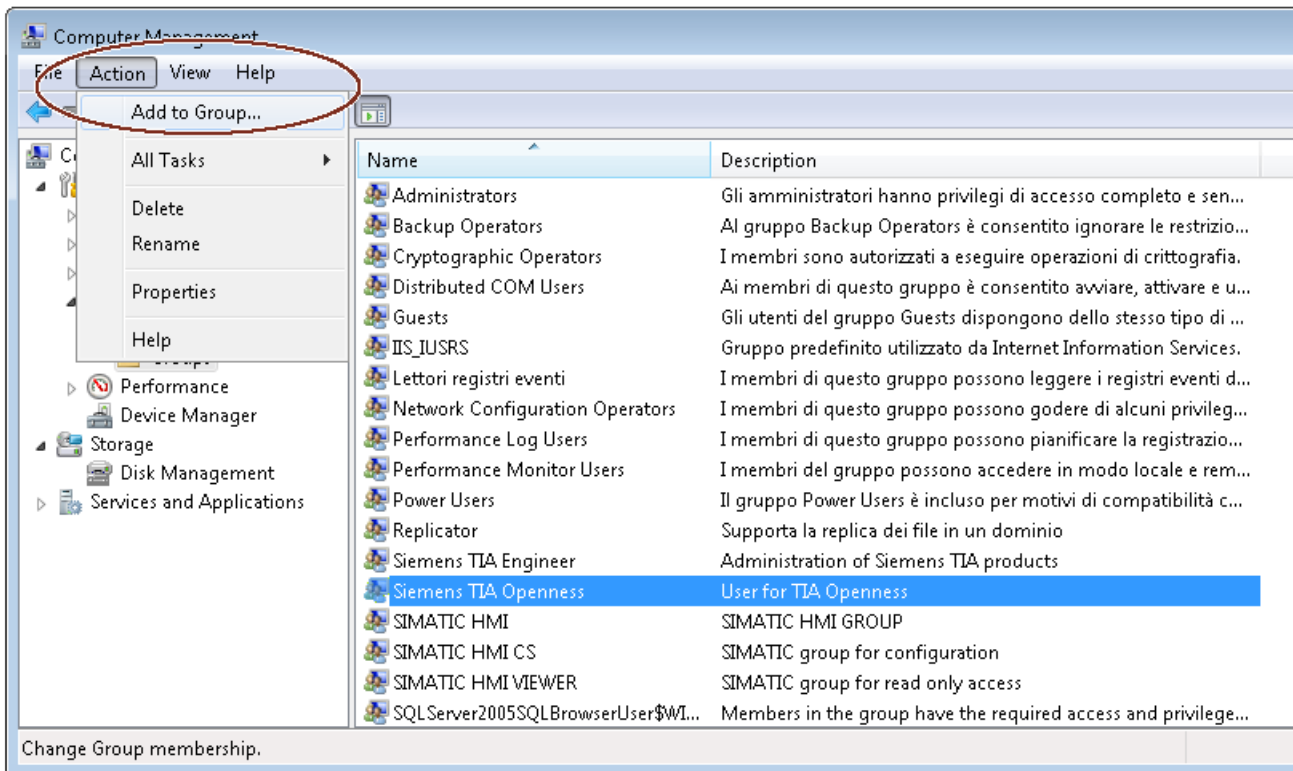


4. Select "Local Users and Groups > Groups", in order to display all created user groups.

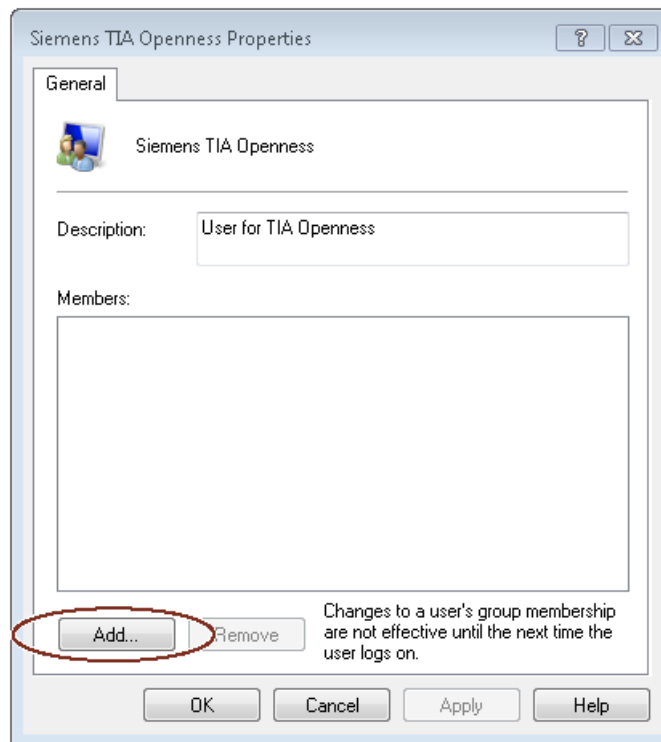
5. Select the "Siemens TIA Openness" entry from the list of user groups in the right pane.



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

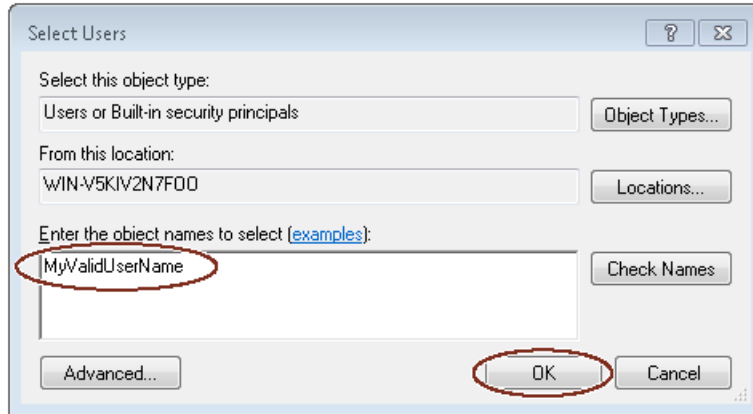


The attributes dialog of the user group opens:



- 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.

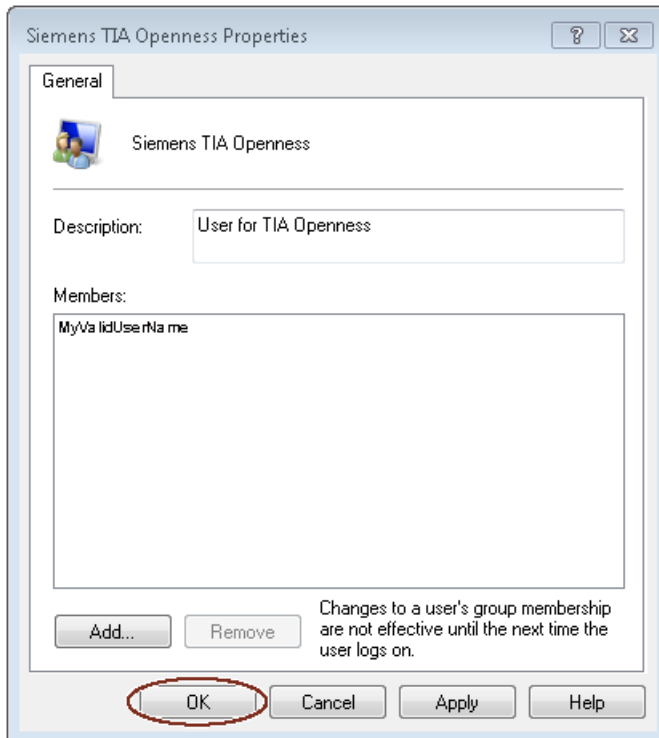
**Note**

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.



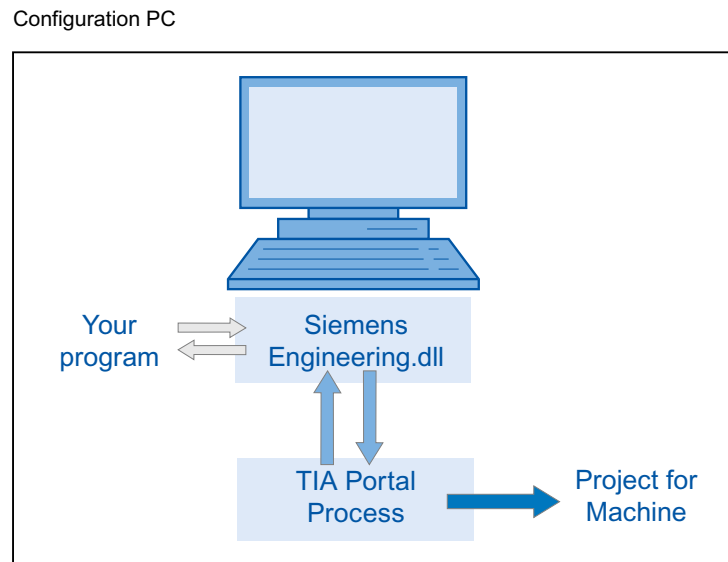
You register additional users by clicking the "Add" button.



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

### 4.2.3 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 74)
- Terminating the connection to the TIA Portal (Page 83)

## 4.3 Openness tasks

### 4.3.1 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 :

- Project data
- PLC data
- HMI data

---

#### Note

You must not use the TIA Portal Openness API to execute checks or generate data for the acceptance/approval of a fail-safe system. Acceptance/approval may only be carried out with a safety printout using the STEP 7 Safety add-on package or with the function test. The TIA Portal Openness API is no substitute.

---

#### 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

## 4.3.2 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

## 4.4 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 4-1 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 4-2 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
T-piece	No	No	No	No

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
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 <sup>1)</sup>	Yes	Yes
ProDiag overview	No	Yes <sup>1)</sup>	Yes	Yes
GRAPH overview	No	Yes <sup>1)</sup>	Yes	Yes
PLC code view	No	Yes <sup>1)</sup>	Yes	Yes
f(x) trend view	No	No	No	No
f(t) trend view	No	No	No	No
Table view	No	No	No	No
Value table	No	No	No	No

4.4 Object list

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
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 4-3 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 4-4 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 4-5 Tags

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Multiplex tags	Yes	Yes	Yes	Yes
Arrays	Yes	Yes	Yes	Yes

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
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 4-6 Lists

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

Table 4-7 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

## 4.5 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 Siemens.Engineering;
using Siemens.Engineering.CAx;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Extension;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.HW.Utilities;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.TechnologicalObjects;
using Siemens.Engineering.SW.TechnologicalObjects.Motion;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
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.Online;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using Siemens.Engineering.Library.Types;
using Siemens.Engineering.Library.MasterCopies;
using Siemens.Engineering.Compare;
using System.IO;
```



## 4.6 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.



# 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](http://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.

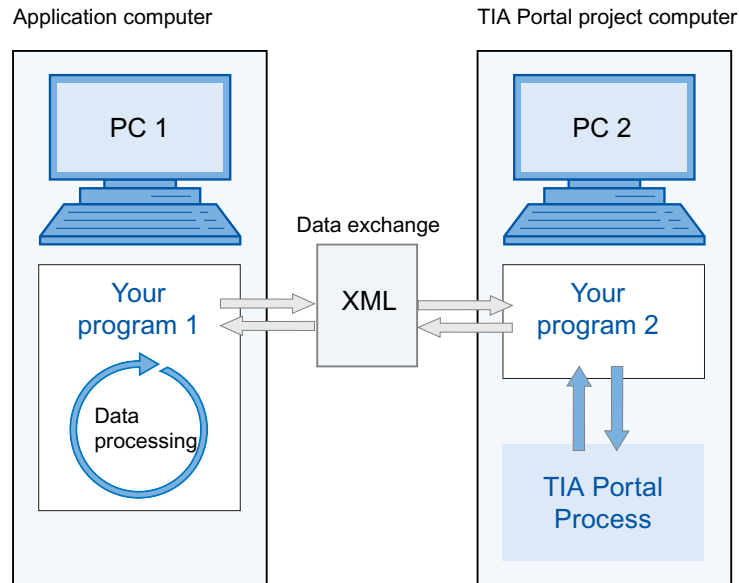
**See also**

Configurations (Page 45)

# 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.

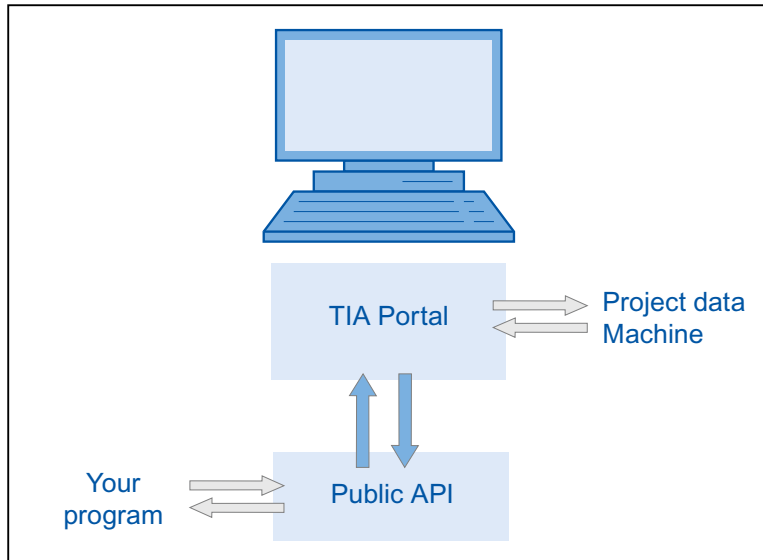
Import and export of XML files takes place exclusively via the TIA Portal Openness API.

---

- 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

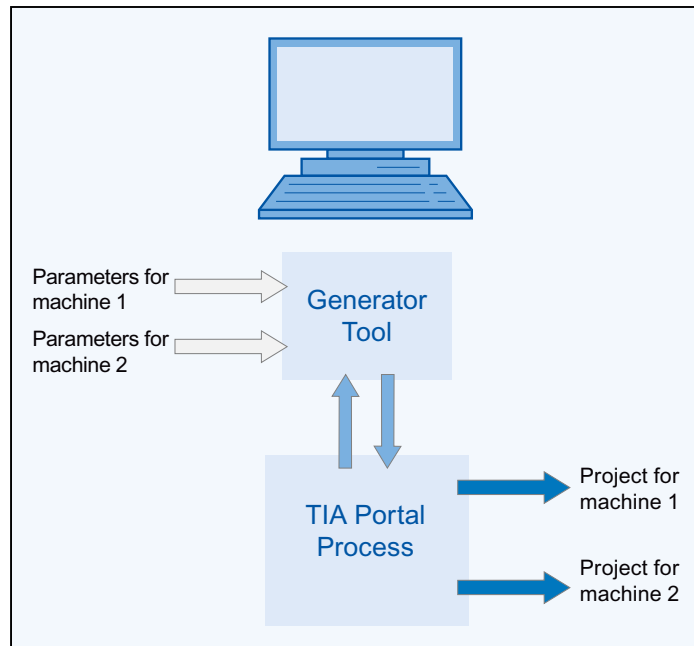
Configuration PC



- 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

Configuration computer



- 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.





# TIA Portal Openness API

## 7.1 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 34)

Object list (Page 36)

## 7.2 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

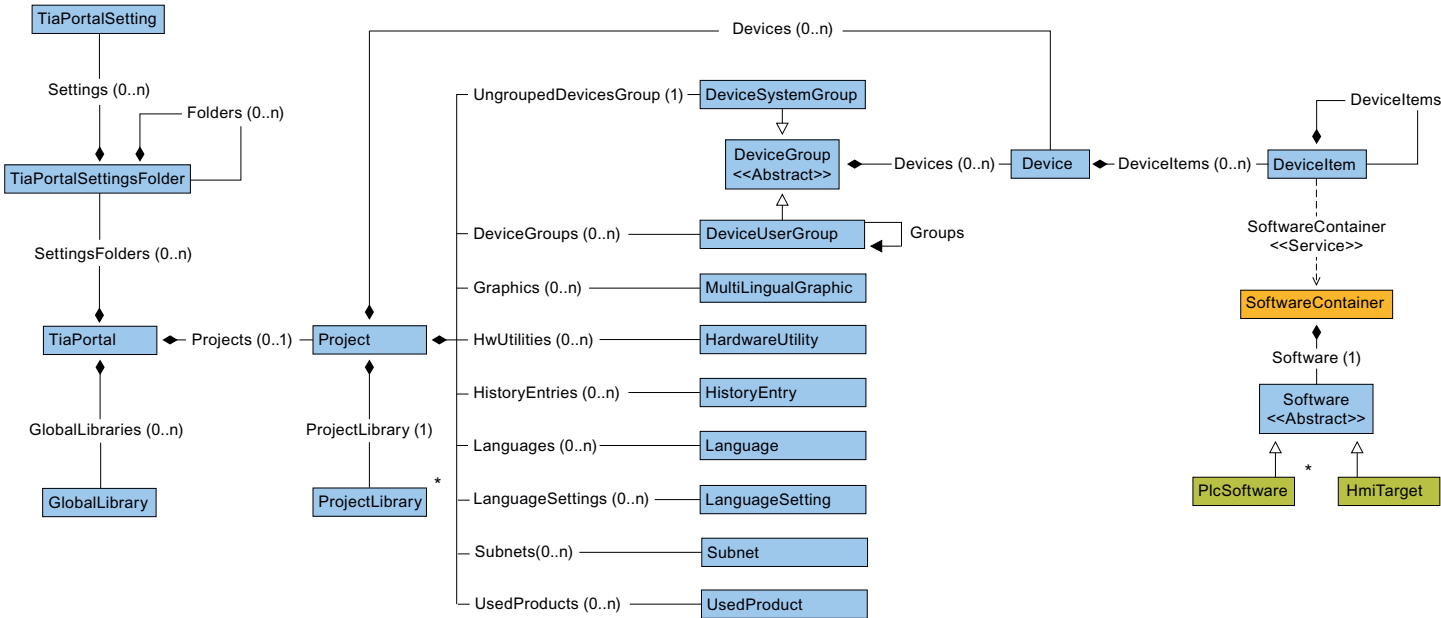
Example program (Page 67)

Use of the code examples (Page 72)

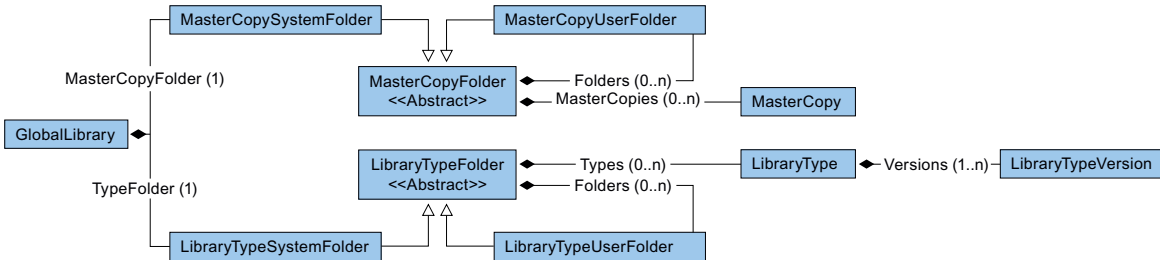
## 7.3 TIA Portal Openness object model

### Overview

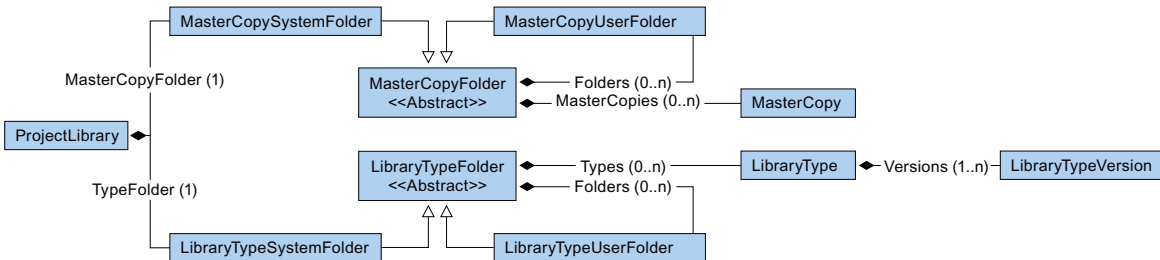
The following diagram describes the highest level of the TIA Portal Openness object model:



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

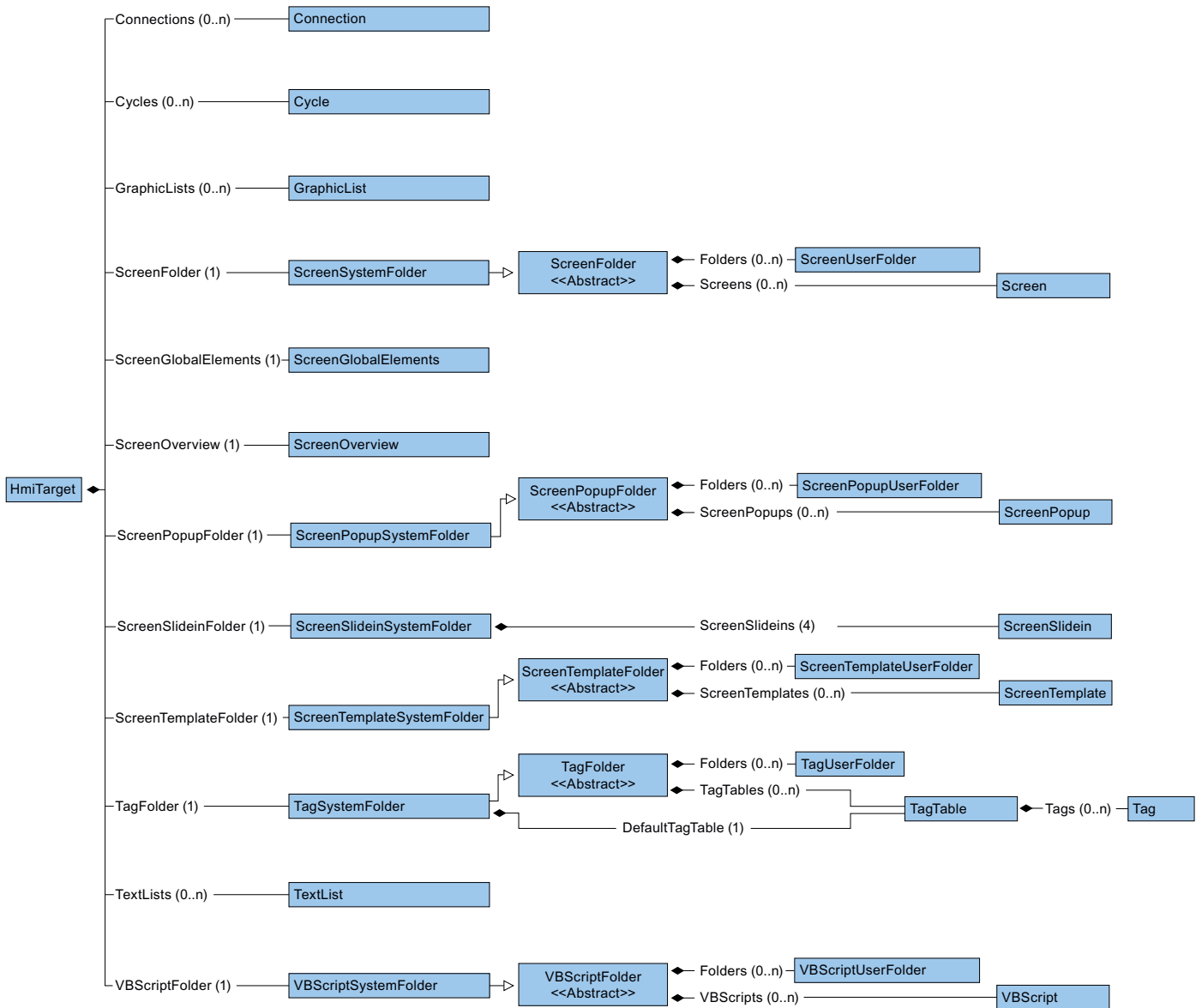


The following diagram describes the objects which are located under ProjectLibrary.



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

7.3 TIA Portal Openness object model



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



## Access to objects in lists

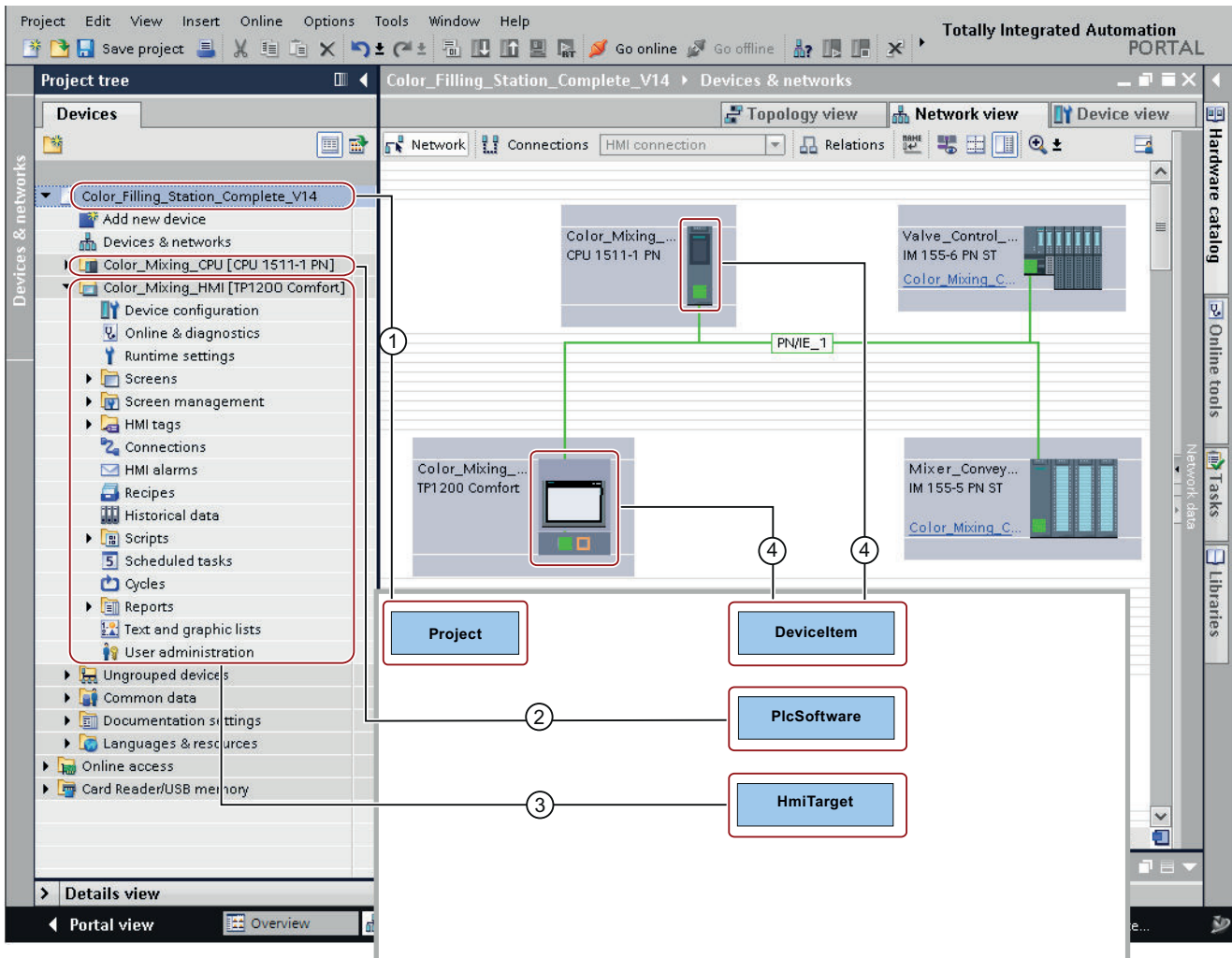
You have the following options for addressing an object in a list:

- 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**

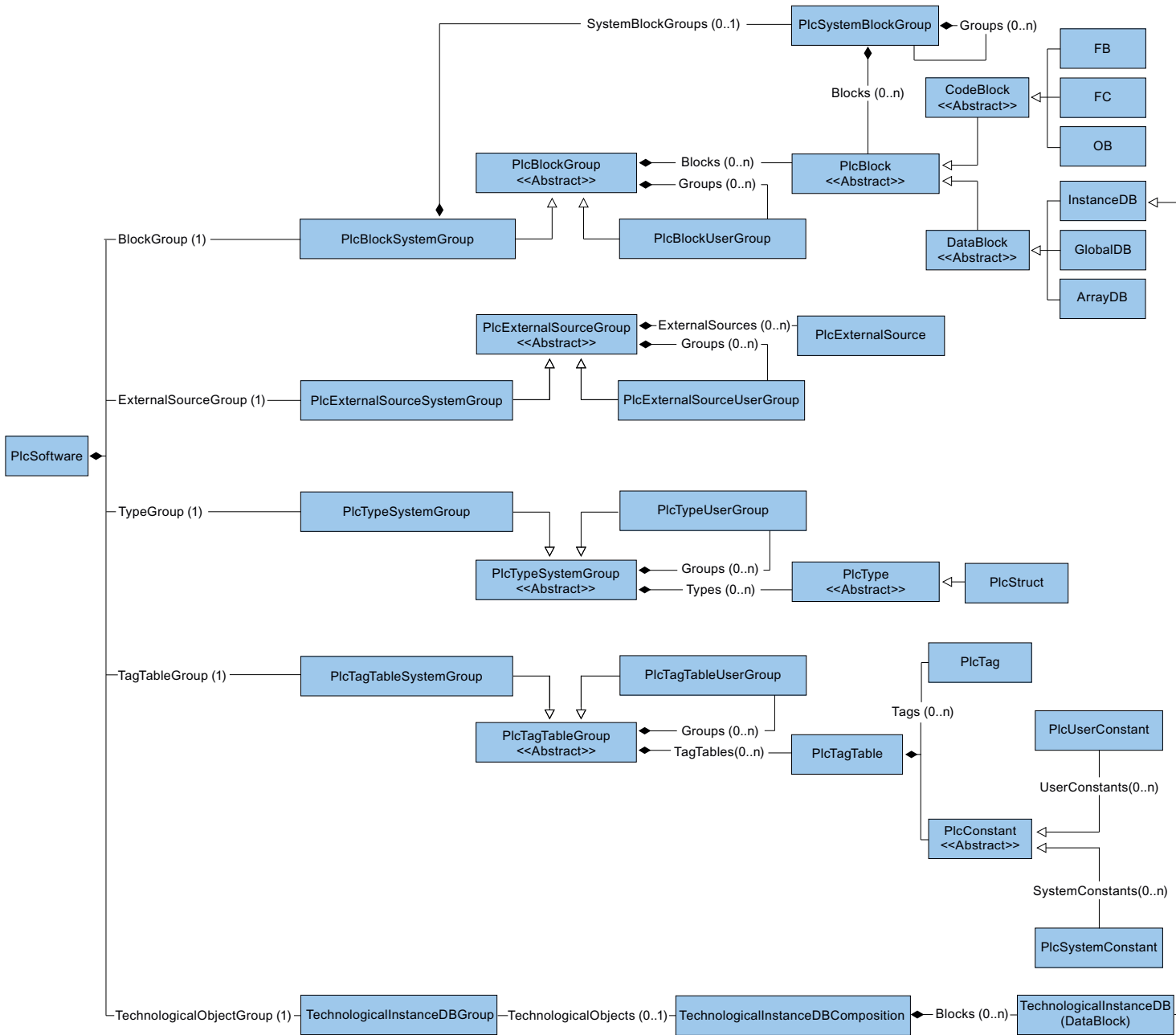
Blocks and types of the TIA Portal Openness object model (Page 56)

Hierarchy of hardware objects of the object model (Page 64)

## 7.4 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.





## **Representation of blocks and types in the TIA Portal Openness API**

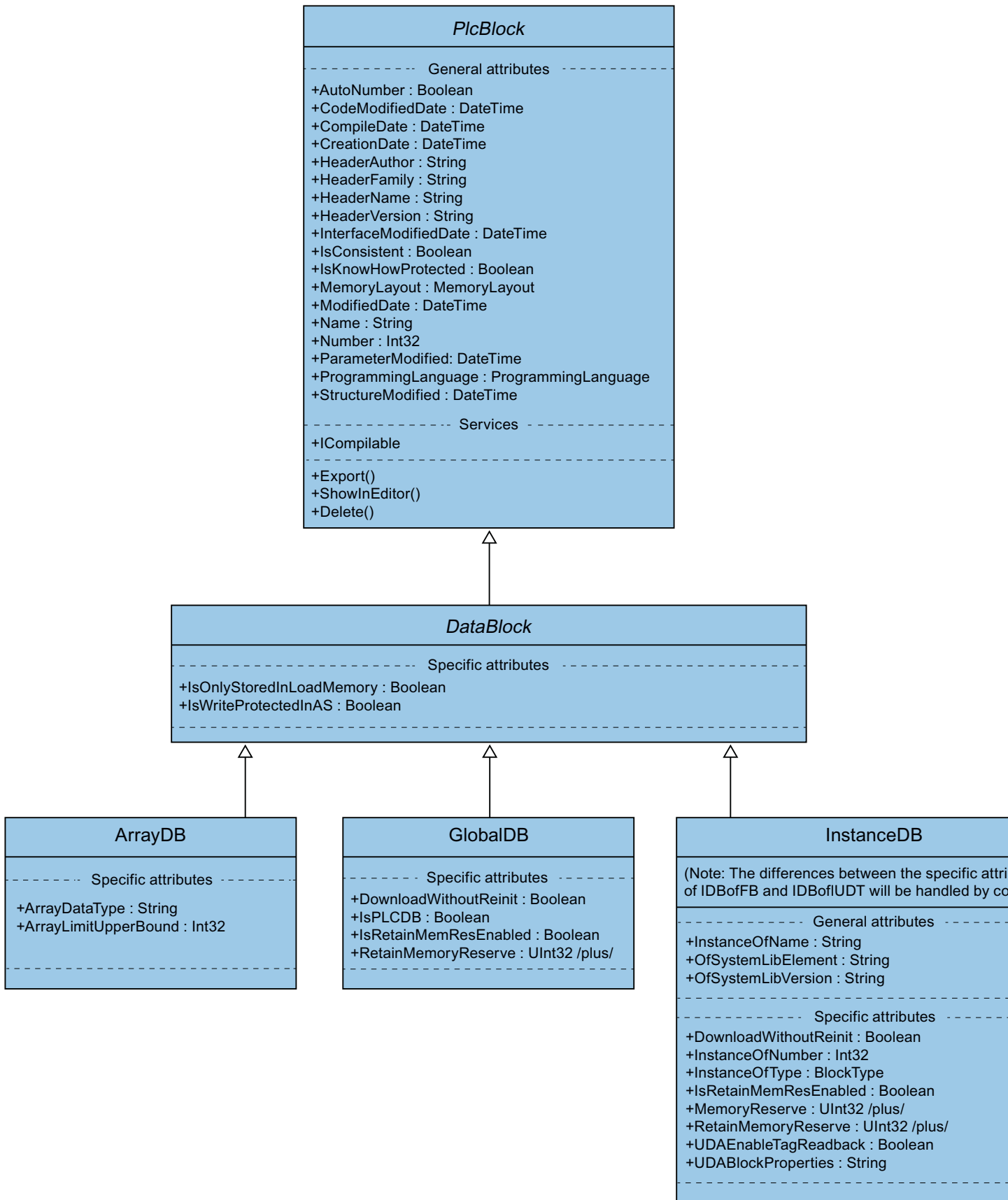
The simplified model part of blocks and for the structure is based on the attributes in the TIA Portal Openness API. These classes provide the export function and for blocks also the compile function.

### **Class diagrams**

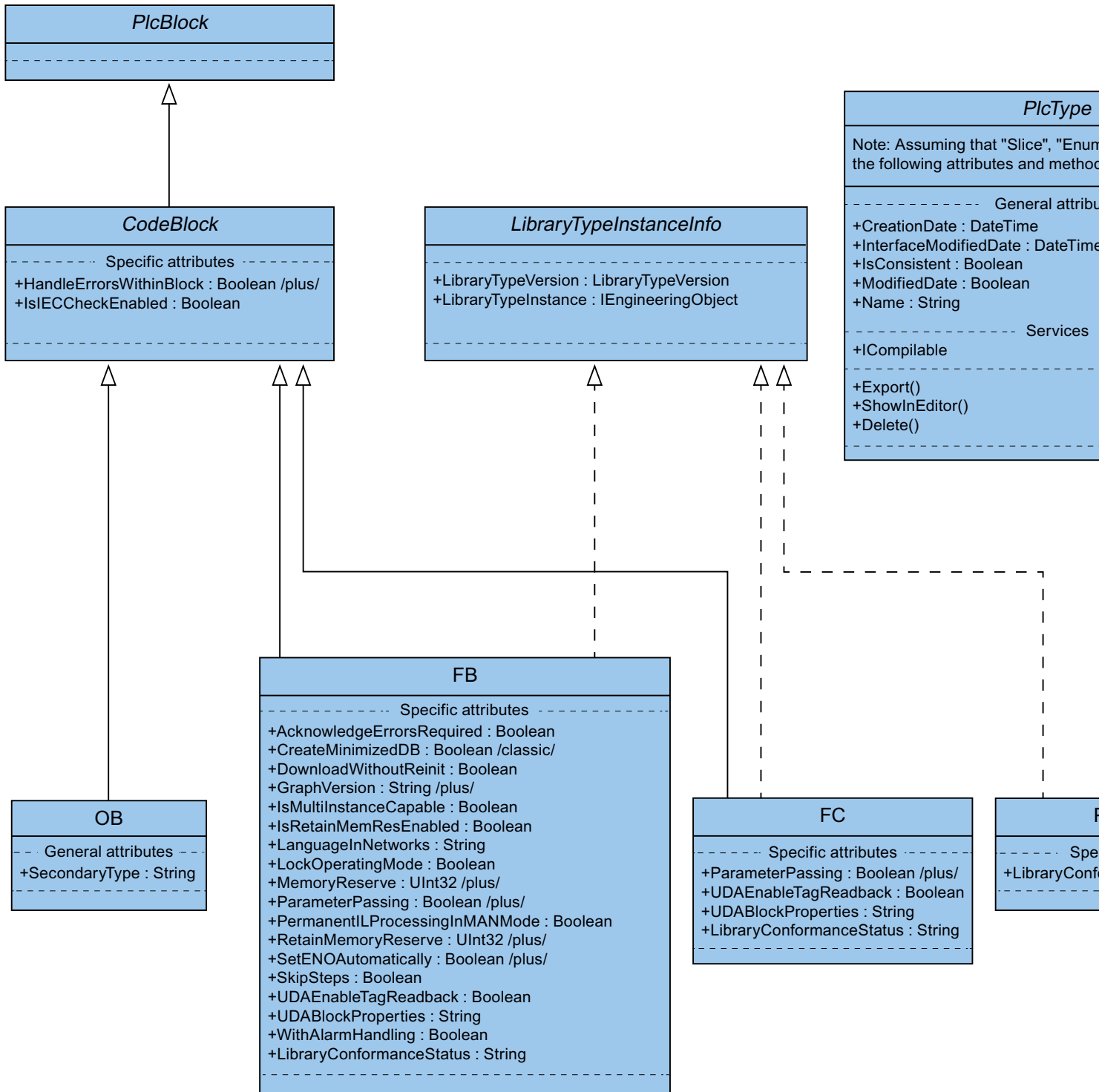
In the TIA Portal Openness object model all classes are defined as abstract which aren't directly instantiated.

**Data**

7.4 Blocks and types of the TIA Portal Openness object model



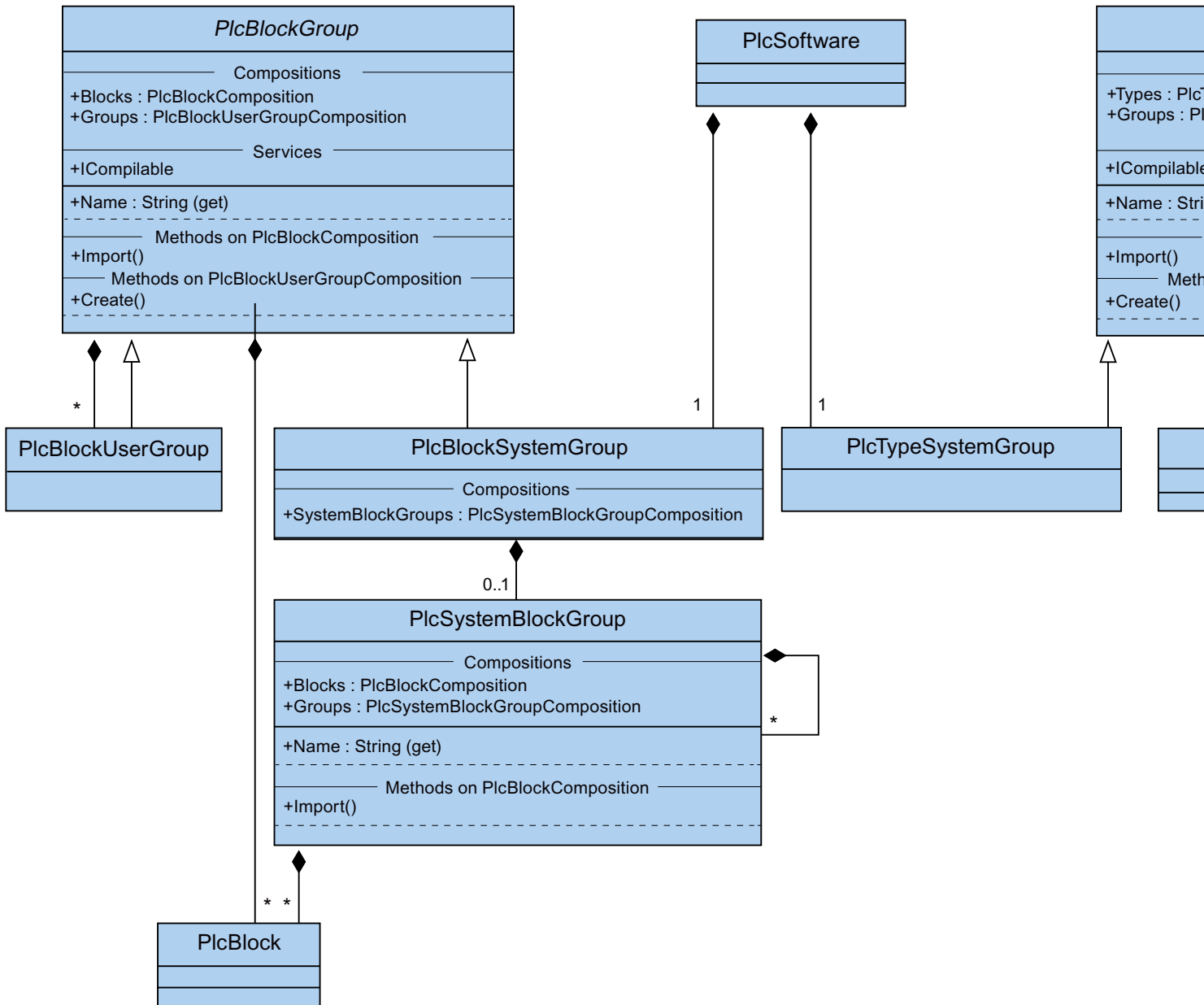
Code and Type



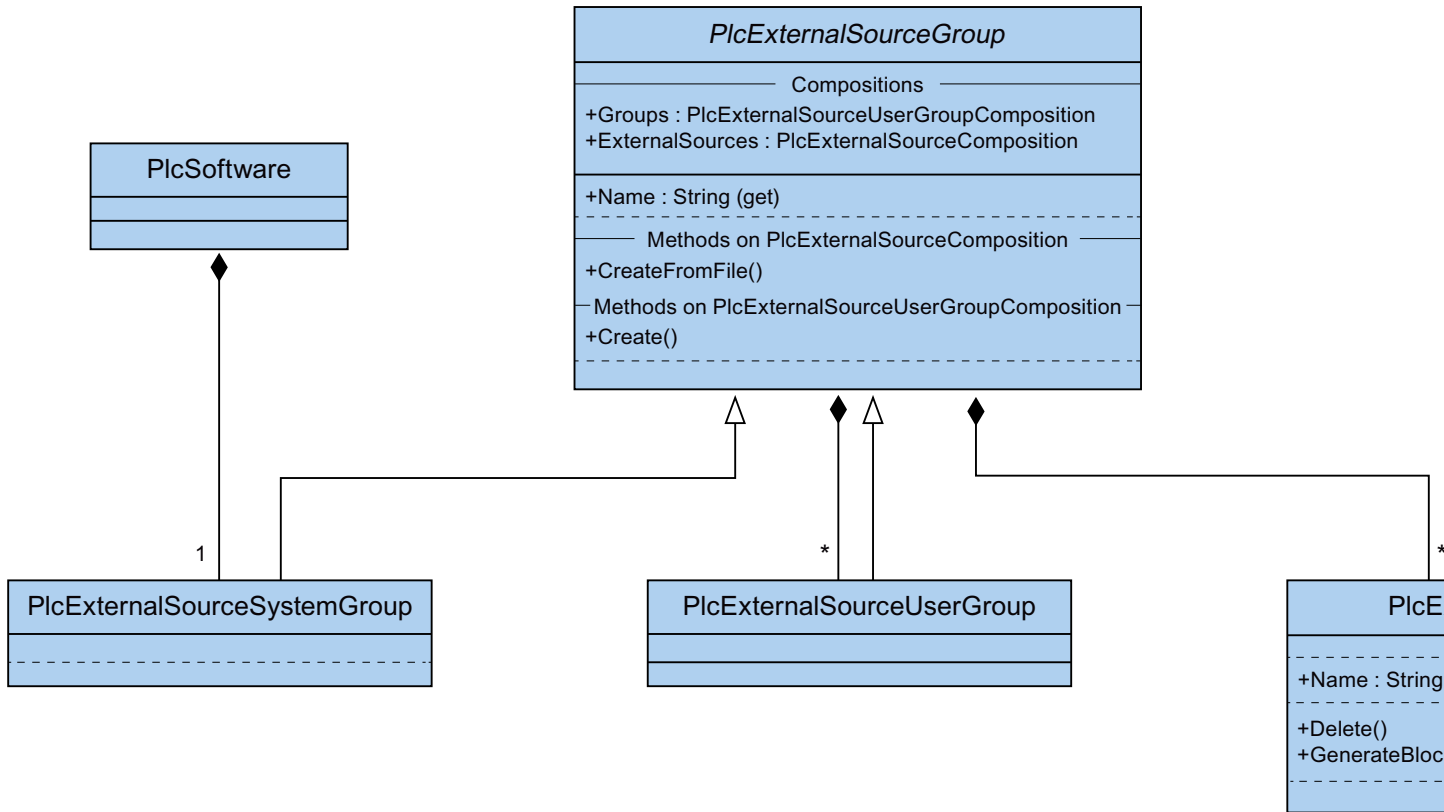
## Representation of groups for blocks and types in the TIA Portal Openness API

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 51)

Hierarchy of hardware objects of the object model (Page 64)

## 7.5 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 (Deviceltem)	Each device item object has a container object. The logical relation is "Items".

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

Example: A device includes one or more slots. A slot includes modules. A module includes submodules.

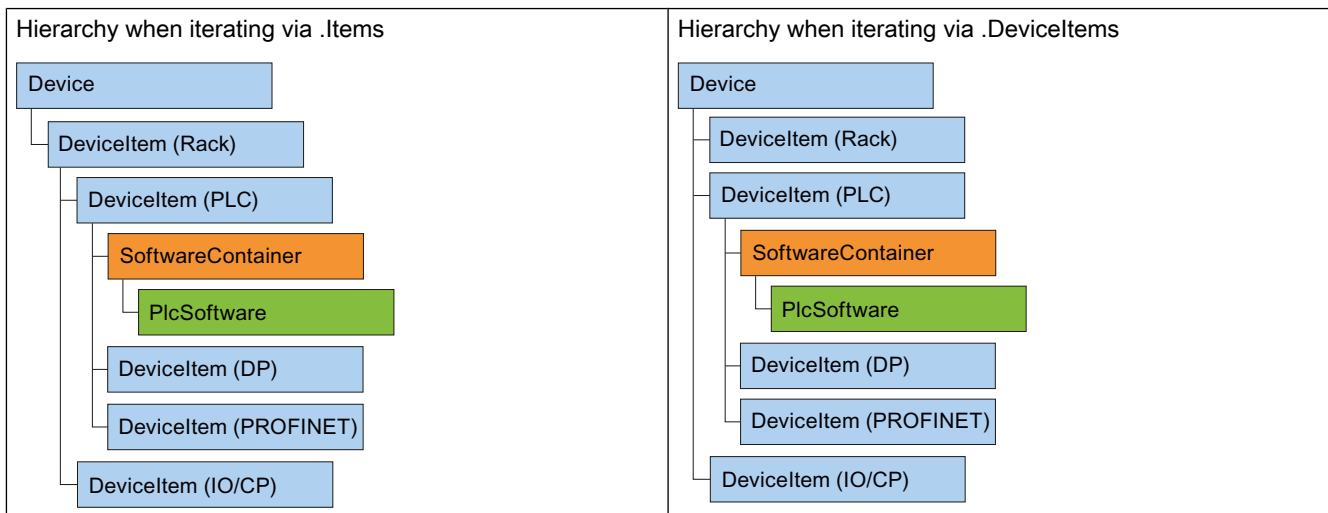
This is the relation similar to the representation in the network view and device view of the TIA Portal. The "PositionNumber" attribute of a device item is unique in the items area, within a container.

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.
- If you add and then remove a submodule from the module, this child has the same parents as the module.

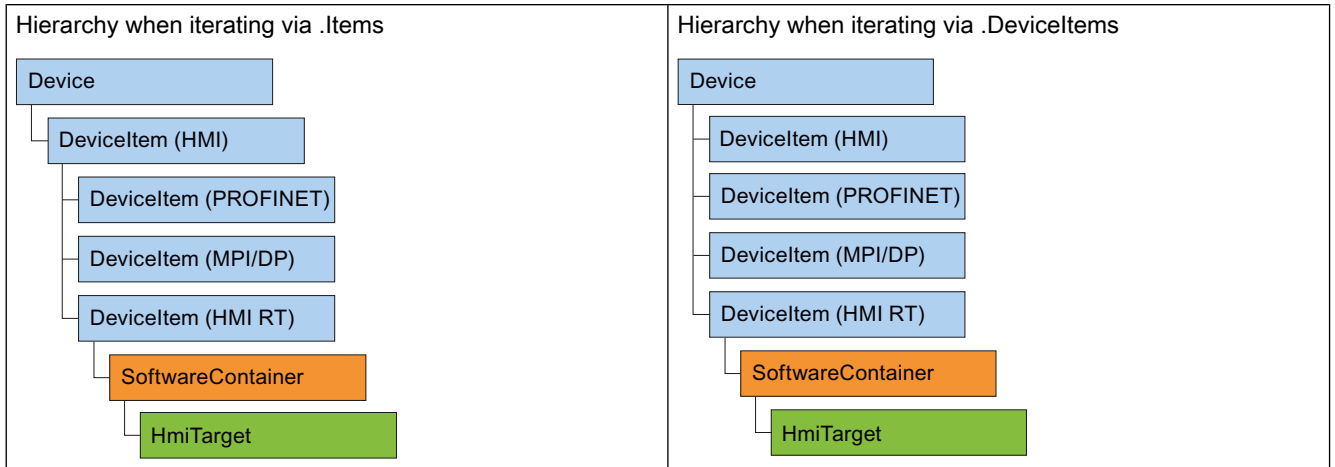
The diagrams below show the hierarchy relationship between devices, and device items of PLC and HMI devices.

### Hierarchy relationships of PLC devices





## Hierarchy relationships of HMI devices



### See also

TIA Portal Openness object model (Page 51)

Blocks and types of the TIA Portal Openness object model (Page 56)

## 7.6 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  
\14.0\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\14.0\PublicAPI  
\14.0.1.0]"PublicKeyToken"="d29ec89bac048f84"  
"Siemens.Engineering"="C:\Program Files\Siemens\Automation\Portal V14\PublicAPI  
\V14\Siemens.Engineering.dll"  
"Siemens.Engineering.Hmi"="C:\Program Files\Siemens\Automation\Portal V14\PublicAPI  
\V14\Siemens.Engineering.Hmi.dll"  
"EngineeringVersion"="V14 SP1"  
"AssemblyVersion"="14.0.1.0"
```

---

#### Note

If you want to generate an app.config file (Page 74) you can get the path of the Siemens.Engineering.dll, the Siemens.Engineering.Hmi.dll and the public key token from the registry key.

---

## 7.7 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 74).

---

---

## 7.7 Example program

```
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...");

                FileInfo projectPath = new FileInfo("C:\\Demo\\AnyCompanyProject.ap14"); //
edit the path according to your project
                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));

    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...");  
FileInfo projectPath = new FileInfo("C:\\Demo\\AnyCompanyProject.ap14");  
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();
```

## 7.8 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();
    }
}
```

## 7.9 General functions

### 7.9.1 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);
```

ProjectProjectComposition.OpenWithUpgrade(FileInfo path)  
Open Action with project update is necessary

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.

### 7.9.2 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 installation -->
        <codeBase version="15.1.0.0" href="FILE:///C:\Program Files\Siemens\Automation
\Portal V15.1\PublicAPI\V15.1\Siemens.Engineering.dll"/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Siemens.Engineering.Hmi" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
        <!-- Edit the following path according to your installation -->
        <codeBase version="15.1.0.0" href="FILE:///C:\Program Files\Siemens\Automation
\Portal V15.1\PublicAPI\V15.1\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 installation
            string path = Path.Combine(@"C:\Program Files\Siemens\Automation\Portal
V14\PublicAPI\V14 SP1\", 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())
{
    //...
}
```

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.

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 79) for further information.

---

### Note

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

---

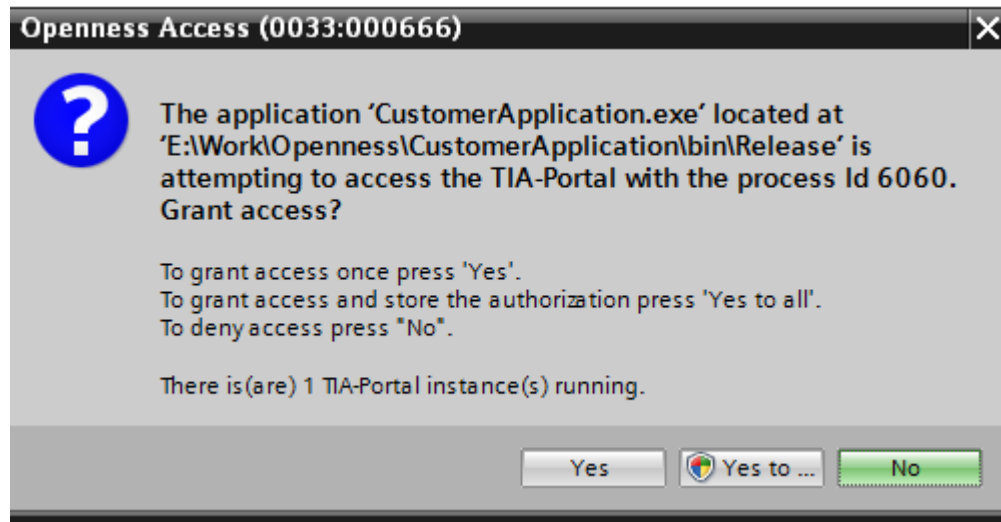
Once you have connected to the process, you can use the following attributes to retrieve information on the instances of the TIA Portal:

Attribute	Information
InstalledSoftware as <code>IList&lt;TiaPortalProduct&gt;</code>	Returns information about the installed products.
Mode as <code>TiaPortalMode</code>	Returns the mode in which the TIA Portal was started ( <code>WithoutUserInterface/WithUserInterface</code> ).
AttachedSessions as <code>IList&lt;TiaPortalSession&gt;</code>	Returns a list of applications connected to the TIA Portal.
ProjectPath as <code>FileInfo</code>	Returns the file name of the project opened in the TIA Portal, including the folder, for example, <code>"D:\WinCCProjects\ColorMixing\ColorMixing.ap14"</code> If no project is open, a null string is returned.
ID as <code>int</code>	Returns the process ID of the TIA Portal instance
Path as <code>FileInfo</code>	Returns the path to the TIA Portal executable

### 7.9.3 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 whitelist 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\14.0\Whitelist\CustomerApplication.exe]
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\14.0\Whitelist\CustomerApplication.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:

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

### 7.9.4 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 82)

## 7.9.5 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 74)
- A project is open.  
See Opening a project (Page 97)
- Event handlers are registered.  
See Connecting to the TIA Portal (Page 74)

### 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 respond to a confirmation event:

```
...
    tiaPortal.Confirmation += TiaPortalConfirmation;
...
private void TiaPortalConfirmation(object sender, ConfirmationEventArgs e)
{
    ...
}
```

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

```
//Handles notifications
using (TiaPortal tiaPortal = new TiaPortal())
{
    tiaPortal.Notification += Notification;
    try
    {
        //perform actions that will result in a notification event
    }
    finally
    {
        tiaPortal.Notification -= Notification;
    }
}
```

## 7.9.6 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 `IDisposable.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 80)

## 7.9.7 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 its 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	ICollection<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.
Dispose	void	Closes the associated TIA Portal instance.
Id	int	The Process ID of the TIA Portal.
InstalledSoftware	ICollection<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.
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	ICollection<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();
    }
}
```

## 7.9.8 Exclusive access

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### 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

---

**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
    ...
}
...
```

## 7.9.9 Transaction handling

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open. See [Opening a project \(Page 97\)](#)

### 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 instantiate 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.

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.

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

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

**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 Close

## Undo behavior

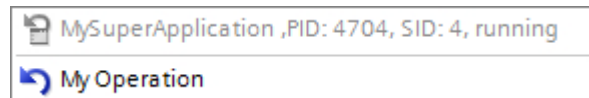
Actions performed by a TIA Portal Openness client application can result in undo units within the attached TIA Portal process. Each of these undo entries will be grouped under a location entry. This location entry will compose the following information from the client application:

- the assembly title from the manifest data if available; otherwise, the process name
- the process ID
- the SID
- optionally an indication that the client process is still running

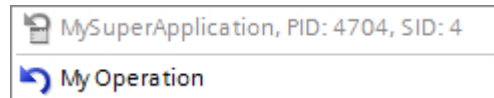
These entries will be one of the following two kinds:

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:

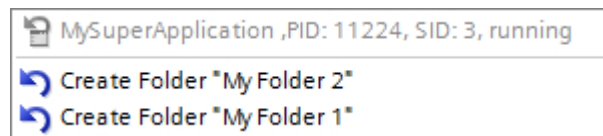


- Undo entry for a stopped client application:

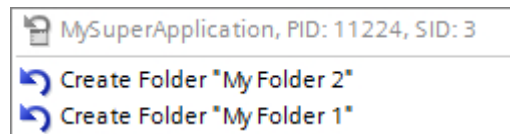


2. The operations that are executed individually have individual undo entries describing the operation as defined in the respective command meta data.

- Undo entry for a running client application:



- Undo entry for a stopped client application:



### 7.9.10 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.

```
..
//Create a DirectoryInfo object
string directoryPath = @"D:\Test\Project 1";
DirectoryInfo directoryInfo = new DirectoryInfo(directoryPath);

//Create a FileInfo object
string fileName = @"D:\Test\Project 1\Project 1.ap14";
FileInfo fileInfo = new FileInfo(fileName);
...
```

### 7.9.11 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
<code>GetCompositionInfos</code>	Returns a collection of <code>EngineeringCompositionInfo</code> objects describing the different compositions of these objects. <code>EngineeringCompositionInfo</code> is described below.
<code>GetAttributeInfos</code>	Returns a collection of <code>EngineeringAttributeInfo</code> objects describing the different attributes of these objects. <code>EngineeringAttributeInfo</code> is described below.
<code>GetInvocationInfos</code>	Returns a collection of <code>EngineeringInvocationInfo</code> objects describing the different actions of these objects. <code>EngineeringInvocationInfo</code> 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.

**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.

**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

## 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;  
    }  
}  
...  
...
```



## 7.10 Functions for projects and project data

### 7.10.1 Opening a project

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- 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 multiple projects

You can open multiple projects in an instance of the TIA Portal. In this case you have to decide to 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. If a project is opened as secondary, the project will not be reflected in the user interface. However, you can always open the UMAC protected project as secondary in read-only mode even with read and write privileges. 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`.

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

## 7.10.2 Creating a project

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)

### 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 project folder.

**7.10.3 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 74)
- A project is open.  
See Opening a project (Page 97)

## Application

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

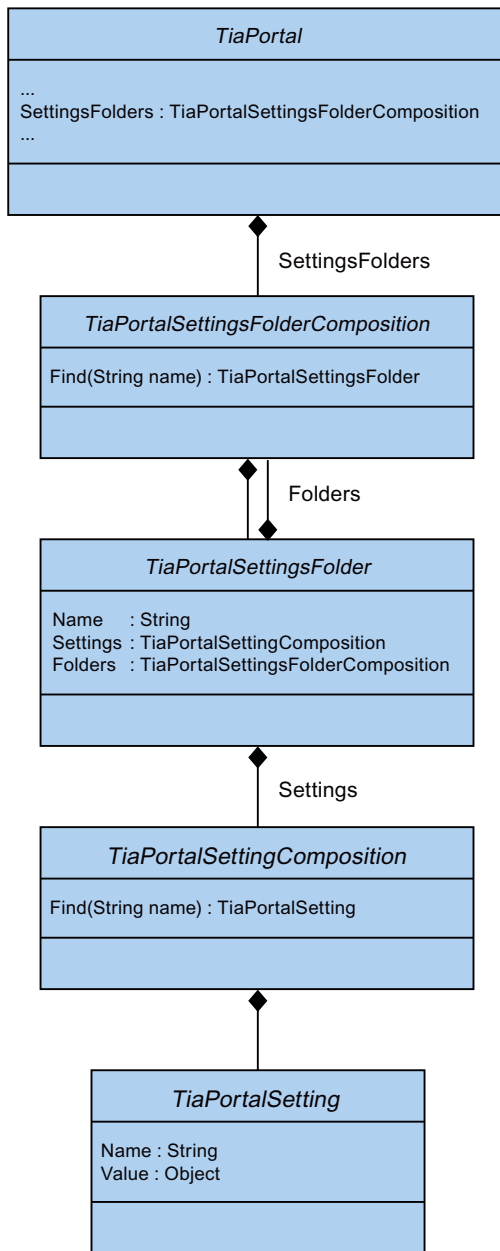
- Current user interface language
- "Search in project" option to create the search index needed for searching within a project.

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:





**Program code: Search in project**

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

```
private static void SetSearchInProject(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**

Hierarchy of hardware objects of the object model (Page 64)

## 7.10.4 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 74)
- A project is open  
See Opening a project (Page 97)

### 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 97). 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 97)

## 7.10.5 Accessing languages

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

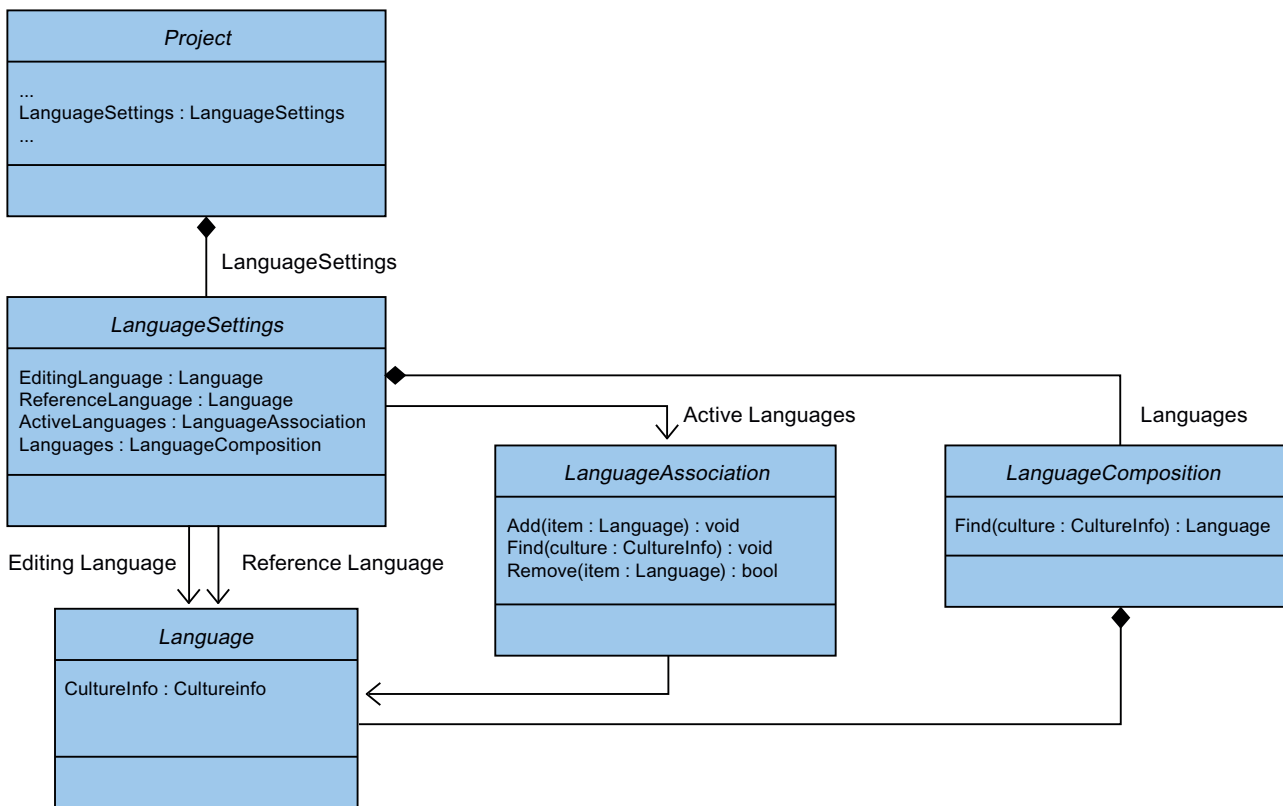
### Application

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.
- 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.

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.

```

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.

```
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 64)

## 7.10.6 Determining the object structure and attributes

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application. See [Opening a project \(Page 97\)](#)

### 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 all 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 DisplayAttributeInfos(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;
}
```

### 7.10.7 Access software target

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

## Programm code

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

```
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;
}
```

## 7.10.8 Accessing and enumerating multilingual texts

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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;
        }
    }
...

```

**7.10.9 Read project related attributes****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

**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 occurred

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

## 7.10 Functions for projects and project data

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;
}
```

### 7.10.10 Deleting project graphics

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open. See [Opening a project \(Page 97\)](#)

#### 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();
}
```

### 7.10.11 Compiling a project

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open. See [Opening a project \(Page 97\)](#)
- 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()`.

---

## 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);
    }
}
```

### See also

Importing configuration data (Page 417)

## 7.10.12 Saving a project

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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 = null;
    //Use the code in the try block to open and save a project
    try
    {
        project = tiaPortal.Projects.Open(new FileInfo(@"Some\Path\MyProject.ap14"));
        //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);
...
```

### 7.10.13 Closing a project

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- You have opened a project with your TIA Portal Openness application. See Opening a project (Page 97)



## Program code

Modify the following program code to close a project:

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

## 7.11 Functions for Connections

### 7.11.1 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 97)
- A project is open.  
See Opening a project (Page 97)

#### 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	MediumAttachmentType	r/o	Dynamic attribute	
CableName	CableName	r/w	Dynamic attribute	
AlternativePartnerPorts	Boolean	r/w	Dynamic attribute	Only available if toolchanger functionality is supported, e.g. at CPU1516.
SignalDelaySelection	SignalDelaySelection	r/w	Dynamic attribute	
CableLength	CableLength	r/w	Dynamic attribute	
SignalDelayTime	Double	r/w	Dynamic attribute	

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

Value	Description
<code>MediumAttachmentType.None</code>	Attachment type cannot be determined.
<code>MediumAttachmentType.Copper</code>	Attachment type is copper.
<code>MediumAttachmentType.FibreOptic</code>	Attachment type is fiber optic.

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

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

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

Value	Description
<code>SignalDelaySelection.None</code>	
<code>SignalDelaySelection.CableLength</code>	<code>CableLength</code> is used to define the signal delay.
<code>SignalDelaySelection.SignalDelayTime</code>	<code>SignalDelayTime</code> is used to define the signal delay.

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

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

## Port options

The following attributes are provided for port options:

Attribute name	Data type	Writeable	Access
<code>PortActivation</code>	<code>bool</code>	r/w	Dynamic attribute
<code>TransmissionRateAndDuplex</code>	<code>TransmissionRateAndDuplex</code>	r/w	Dynamic attribute
<code>PortMonitoring</code>	<code>bool</code>	r/w	Dynamic attribute

Attribute name	Data type	Writeable	Access
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
<code>TransmissionRateAndDuplex.None</code>	
<code>TransmissionRateAndDuplex.Automatic</code>	Automatic
<code>TransmissionRateAndDuplex.AUI10Mbps</code>	10 Mbps AUI
<code>TransmissionRateAndDuplex.TP10MbpsHalfDuplex</code>	TP 10 Mbps half duplex
<code>TransmissionRateAndDuplex.TP10MbpsFullDuplex</code>	TP 10 Mbps full duplex
<code>TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex</code>	async fiber 10Mbit/s half duplex mode
<code>TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex</code>	async fiber 10Mbit/s full duplex mode
<code>TransmissionRateAndDuplex.TP100MbpsHalfDuplex</code>	TP 100 Mbps half duplex
<code>TransmissionRateAndDuplex.TP100MbpsFullDuplex</code>	TP 100 Mbps full duplex
<code>TransmissionRateAndDuplex.FO100MbpsFullDuplex</code>	FO 100 Mbps full duplex
<code>TransmissionRateAndDuplex.X1000MbpsFullDuplex</code>	X1000 Mbps full Duplex
<code>TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD</code>	FO 1000 Mbps full duplex LD
<code>TransmissionRateAndDuplex.FO1000MbpsFullDuplex</code>	FO 1000 Mbps full Duplex
<code>TransmissionRateAndDuplex.TP1000MbpsFullDuplex</code>	TP 1000 Mbps full duplex
<code>TransmissionRateAndDuplex.FO10000MbpsFullDuplex</code>	FO 10000 Mbps full Duplex
<code>TransmissionRateAndDuplex.FO100MbpsFullDuplexLD</code>	FO 100 Mbps full duplex LD
<code>TransmissionRateAndDuplex.POFPCF100MbpsFullDuplexLD</code>	POF/PCF 100 Mbps full duplex

## See also

Connecting to the TIA Portal (Page 74)

## **7.12 Functions on libraries**

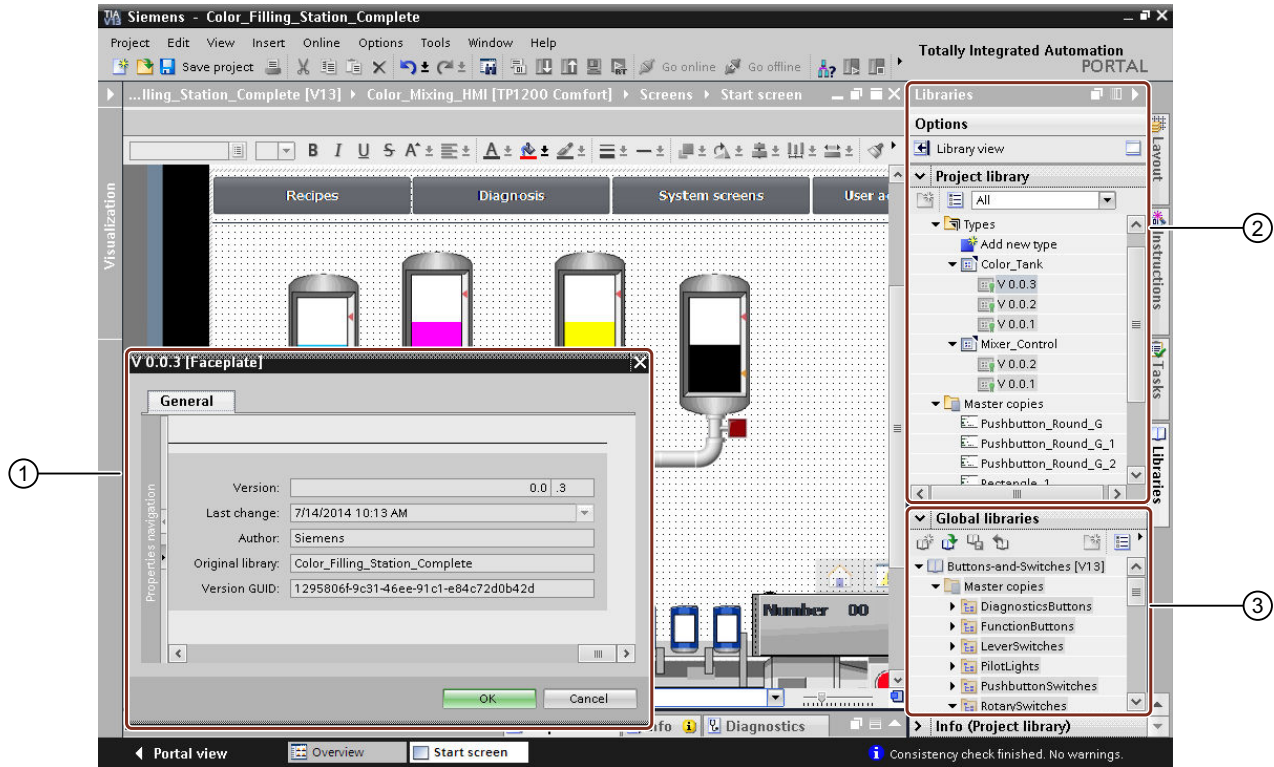
### **7.12.1 Functions for objects and instances**

#### **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

### 7.12.2 Accessing global libraries

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)

## Application

Three types of Global Libraries are existing.

- **System global library:** These global libraries are included as part of a TIA Portal installation and use the .as14 file extension. All system global libraries are read only.
- **Corporate global library:** 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:** 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 opened in a certain mode then the same user global library cannot be opened 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)
{
    //work with the global library info
    Console.WriteLine("Library Name: ", info.Name);
    Console.WriteLine("Library Path: ", info.Path);
    Console.WriteLine("Library Type: ", info.LibraryType);
    Console.WriteLine("Library IsOpen: ", 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: <ul style="list-style-type: none"> <li>• System: System global library</li> <li>• Corporate: Corporate global library</li> <li>• User: User global library</li> </ul>
Name	String	Name of the global library.
Path	FileInfo	Path of the global library.

### See also

Accessing folders in a library (Page 137)

## 7.12.3 Accessing global library languages

### Requirements

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A library is open. See Opening libraries (Page 130)

### 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: Accessing language settings

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

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

## Program code: Enumerating global library language

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

```
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.

```
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.

---

## 7.12.4 Opening libraries

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application. This requirement is only for accessing project libraries. See [Opening a project \(Page 97\)](#)

### 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 can 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.

**7.12.5 Enumerating open libraries****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)

**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 97)

## 7.12.6 Saving and closing libraries

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A library is open.  
See Opening libraries (Page 130)

### 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

## 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;
FileInfo existingLibraryfileInfo = new FileInfo(@"D:\GlobalLibraries\MyGlobalLibrary\MyGlobalLibrary.all15");
DirectoryInfo targetDirectoryInfo = new DirectoryInfo(@"D:\GlobalLibraries\GlobalLibrarySaveAs");
UserGlobalLibrary userGlobalLibrary =
globalLibraryComposition.Open(existingLibraryfileInfo, OpenMode.ReadWrite);
userGlobalLibrary.SaveAs(targetDirectoryInfo);
```

Modify the following program code to close a user global library:

```
UserGlobalLibrary userLib = ...
// close and discard changes
userLib.Close();
```

## 7.12.7 Archiving and retrieving a library

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A library is open. See Opening libraries (Page 130)
- A library is save See Saving and closing libraries (Page 132)

### Application

A opened and saved global libraries can be archived prior to any further modification to prevent from unintended result so that you can later 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.

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

'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 "zal15" 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 would throw an EngineeringTargetInvocationException.

---

### Library Archivation Mode

The LibraryArchivationMode enumeration have four values:

LibraryArchivation-Mode	Description
None	<ul style="list-style-type: none"> <li>No special action are taken with the orginial files. Mode is similiar to a "save as" operation.</li> <li>No compressed zip file is created in this mode.</li> <li>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.</li> </ul>
DiscardRestorable-Data	<ul style="list-style-type: none"> <li>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.</li> <li>No compressed zip file is created in this mode.</li> </ul>
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.
DiscardRestorable-DataAndCompress-ed	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);
var libraryFilePath = @"E:\Sample1\Sample1.all5";
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.

```
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	<ul style="list-style-type: none"> <li>Read access to the library. Data can be read from the library.</li> </ul>
ReadWrite	<ul style="list-style-type: none"> <li>Write access to the library. Data can be written to the library.</li> </ul>

## Program code: Retrieving a library

Modify the following program code to access to retrieve a library:

```
var archivePath = @"E:\Archive\Sample1.zal15";
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 \(Page 130\)](#)

[Saving and closing libraries \(Page 132\)](#)

## 7.12.8 Creating global libraries

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)

### Application

Global libraries can be created via TIA Portal Openness API by calling the Create method on the GlobalLibraryComposition. A UserGlobalLibrary will be returned

### 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.a1XX" will be created

### 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.

### See also

Opening a project (Page 97)



## 7.12.9 Accessing folders in a library

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application.  
See [Opening a project \(Page 97\)](#)
- You have access to the required library.  
See [Accessing global libraries \(Page 126\)](#).

### Application

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

Modify the following program code:

```
...
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 \(Page 148\)](#)

## 7.12.10 Accessing types

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- You have opened a project with your TIA Portal Openness application. See Opening a project (Page 97)
- You have access to the required library. See Accessing global libraries (Page 126).
- You have access to a group for types. See Accessing folders in a library (Page 137).

### 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. <sup>1</sup>
Name	String	Returns the name of the type. <sup>2</sup>

<sup>1</sup> You can find an individual type in a library using this attribute. The search is recursive.

<sup>2</sup> 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.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

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;
```

## 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")});
```

### 7.12.11 Accessing type versions

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application. See [Opening a project \(Page 97\)](#)
- You have access to the required library. See [Accessing global libraries \(Page 126\)](#).
- You have access to a group for types. See [Accessing folders in a library \(Page 137\)](#).

#### Application

You can access type versions via the TIA Portal Openness API interface.

- You can enumerate the type versions of a type.
- You can determine the type to which a type version belongs.
- You can enumerate the instances of a type version.
- You can create a new instance of a type version.
- You can 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. <sup>1</sup>
ModifiedDate	DateTime	Returns the date and time at which the type version was set to the "Committed" status.
State	LibraryTypeVersion-State	Returns the status of the version: <ul style="list-style-type: none"> <li>• <i>InWork</i>: Corresponds to the status "In progress" or "In testing" depending on the associated type.</li> <li>• <i>Committed</i>: Corresponds to the status "Released".</li> </ul>
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". <sup>2</sup>

<sup>1</sup> You can find an individual type version in a library using this attribute.

<sup>2</sup> 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:

```
//Accessing 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 plcTypVersion=...;
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);
}
```

## 7.12.12 Accessing instances

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application.  
See [Opening a project \(Page 97\)](#)
- You have access to the required library.  
See [Accessing global libraries \(Page 126\)](#).
- You have access to a group for types.  
See [Accessing folders in a library \(Page 137\)](#).

### 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 `LibraryTypeInfo` 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<LibraryTypeInfo> instanceInfos = version.FindInstances(searchScope);
IEnumerable<IEngineeringObject> instances = instanceInfos.Select(instanceInfo =>
instanceInfo.LibraryTypeInstance);
```

## Navigate from an instance to its connected version object

Use the `LibraryTypeInfo` attribute of `LibraryTypeInfo` service to navigate from an instance to its connected version object.

The following objects provide the `LibraryTypeInfo` 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
}
```

### Program code

Modify the following program code to .

### 7.12.13 Accessing master copies

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- You have access to the required library.  
See Accessing global libraries (Page 126)
- You have access to a group for master copies.  
See Accessing folders in a library (Page 137)

#### Application

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	MasterCopyContentDescriptionComposition	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;
```

## 7.12.14 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 74)
- A project is open.  
See Opening a project (Page 97)

### 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
- 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);
}
```

### 7.12.15 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 74)
- A project is open.  
See Opening a project (Page 97)
- PLC is not online.

#### Application

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 148)

## 7.12.16 Copying master copies

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### Application

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. Then, it will 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:

```
ProjectLibrary projectLibrary = ...;

MasterCopy copiedMasterCopy =
projectLibrary.MasterCopyFolder.MasterCopies.CreateFrom(sampleMasterCopy)
```

## See also

Accessing master copies (Page 148)

## 7.12.17 Determining out-of-date type instances

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- You have access to the required library.  
See [Accessing global libraries \(Page 126\)](#)
- You have access to a folder for types.  
See [Accessing folders in a library \(Page 137\)](#).

### 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: <ul style="list-style-type: none"> <li>ReportOutOfDateOnly: Returns only status of the "out of date" type.</li> <li>ReportOutOfDateAndUpToDate: Returns status of the type "out of date" and "up to date".</li> </ul>

## 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);
    }
}
```

## 7.12.18 Updating the project

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application.  
See [Opening a project \(Page 97\)](#)
- You have access to the required library.  
See [Accessing global libraries \(Page 126\)](#).
- You have access to a folder for types.  
See [Accessing folders in a library \(Page 137\)](#).

### 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
<code>IEnumerable&lt;ILibraryTypeOrFolderSelection&gt; selectedTypesOrFolders</code>	Specifies the folder or types to be synchronized or their instances in the project to be updated.
<code>IUpdateProjectScope updateProjectScope</code> <code>IEnumerable &lt;IUpdateProjectScope&gt; updateProjectScope</code>	Specifies the object(s) in the project in which the uses of instances are to be updated. The following objects are supported: <ul style="list-style-type: none"> <li>• <code>PlcSoftware</code></li> <li>• <code>HmiTarget</code></li> </ul>

## 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);
}
```

## 7.12.19 Updating a library

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application.  
See [Opening a project \(Page 97\)](#)
- You have access to the required library.  
See [Accessing global libraries \(Page 126\)](#).
- You have access to a folder for types.  
See [Accessing folders in a library \(Page 137\)](#).

### 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
<code>IEnumerable&lt;ILibraryTypeOrFolderSelection&gt; selectedTypesOrFolders</code>	Specifies the folder or types to be synchronized or their instances in the project to be updated.
<code>ILibrary targetLibrary</code>	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);
```

## 7.12.20 Deleting library content

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- You have opened a project with your TIA Portal Openness application. See Opening a project (Page 97)
- You have access to the required library. See Accessing global libraries (Page 126).
- You have access to a folder for types. See Accessing folders in a library (Page 137).

### Application

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 \(Page 148\)](#)

## 7.13 Functions for accessing devices, networks and connections

### 7.13.1 Open the "Devices & networks" editor

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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 417)

**7.13.2 Querying PLC and HMI targets****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

**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 214)

**7.13.3 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 74)
- A project is open.  
See Opening a project (Page 97)
- 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
IsynchronousMode	BOOL	r/w	Dynamic attribute	Activate/Deactivate isochronousMode
ProcessImage	Int32	r/w	Dynamic attribute	Set/Get process image partition number.

Attribute name	Data type	Writeable	Access	Description
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.

## Restrictions

- Attribute `StartAddress`
  - Setting `StartAddress` may implicit change the `StartAddress` of the opposite I/O 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("InterruptObNumber");

// write attribute
address.SetAttribute("InterruptObNumber", 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);
}
```



## 7.13.4 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 74)
- A project is open.  
See Opening a project (Page 97)

### 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.

### 7.13.5 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.

<sup>1</sup>: Not supported by all associations.

## 7.13.6 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 417)" ("None", "Overwrite") by which the user controls the import behavior.

<sup>1</sup>: Not supported by all compositions.

## 7.13.7 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");
}
```

## 7.13.8 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 94) 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.

---

## 7.14 Functions on networks

### 7.14.1 Creating a subnet

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### 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 an interface

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

## 7.14.2 Accessing subnets

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### 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

### 7.14.3 Accessing internal subnets

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### 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;
}
```

**7.14.4 Get type identifier of subnets****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

**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
PROFIBUS-Integrated	Subnet.ProfibusIntegrated
PC-Internal	<i>null</i>

## 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;
```

## 7.14.5 Accessing attributes of a subnet

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### Application

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	Writa-ble	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r	dynam-ic	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	Writ-able	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r/w	dynam-ic	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	dynam-ic	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	Writ-able	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r/w	dynam-ic	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	dynam-ic	Highest MPI address at subnet.
TransmissionSpeed	BaudRate	r/w	dynam-ic	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	Writ-able	Access	Description
Name	string	r		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r	dynam-ic	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	Writ-able	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet

Attribute	Data type	Writ-able	Access	Description
SubnetId	string	r/w	dynam-ic	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	dynam-ic	Highest PROFIBUS address at subnet.
TransmissionSpeed	BaudRate	r/w	dynam-ic	True if the subnet is a default subnet. There is at most one default subnet in a project.
BusProfile	BusProfile	r/w	dynam-ic	The PROFIBUS profile.
PbCableConfiguration	bool	r/w	dynam-ic	True to enable additional PROFIBUS network settings
PbRepeaterCount	int	r/w	dynam-ic	Number of repeaters for copper cable
PbCopperCableLength	double	r/w	dynam-ic	The length of the copper cable
PbOpticalComponentCount	int	r/w	dynam-ic	Number of OLMs and OBTs of fiber-optical cable.
PbOpticalCableLength	double	r/w	dynam-ic	The length of the fiber-optical cable for the PROFIBUS network in km.
PbOpticalRing	bool	r/w	dynam-ic	True if bus parameter are adapted for an optical ring
PbOlmP12	bool	r/w	dynam-ic	True if OLM/P12 is enabled for bus parameter calculation
PbOlmG12	bool	r/w	dynam-ic	True if OLM/G12 is enabled for bus parameter calculation
PbOlmG12Eec	bool	r/w	dynam-ic	True if OLM/G12-EEC is enabled for bus parameter calculation
PbOlmG121300	bool	r/w	dynam-ic	True if OLM/G12-1300 is enabled for bus parameter calculation
PbAdditionalNetworkDevices	bool	r/w	dynam-ic	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	dynam-ic	Number of unconfigured DP masters.
PbTotalDpMaster	int	r	dynam-ic	Number of total DP masters
PbAdditionalPassiveDevice	int	r/w	dynam-ic	Number of unconfigured DP slaves or passive devices.
PbTotalPassiveDevice	int	r	dynam-ic	Number of total DP slaves or passive devices.
PbAdditionalActiveDevice	int	r/w	dynam-ic	Number of unconfigured active devices with FDL/FMS/S/ communication load.
PbTotalActiveDevice	int	r	dynam-ic	Number of total active devices with FDL/FMS/S/ communication load.
PbAdditionalCommunicationLoad	CommunicationLoad	r/w	dynam-ic	Rough quantification of the communication load

Attribute	Data type	Writ-able	Access	Description
PbDirectDataExchange	bool	r/w	dynam-ic	Optimization for direct data exchange.
PbMinimizeTslotForSlaveFailure	bool	r/w	dynam-ic	Minimization for time allocation for slave failure.
PbOptimizeCableConfiguration	bool	r/w	dynam-ic	Optimiazation of the cable configuration.
PbCyclicDistribution	bool	r/w	dynam-ic	True if enable cyclic distribution of bus parame-ter.
PbTslotInit	int	r/w	dynam-ic	Default value of Tslot.
PbTslot	int	r	dynam-ic	Waiting to receive time (slot time)
PbMinTsdr	int	r/w	dynam-ic	Minimum protocol processing time
PbMaxTsdr	int	r/w	dynam-ic	Maximum protocol processing time
PbTid1	int	r	dynam-ic	Idle time 1
PbTid2	int	r	dynam-ic	Idle time 2
PbTrdy	int	r	dynam-ic	Ready time
PbTset	int	r/w	dynam-ic	Setup time
PbTqui	int	r/w	dynam-ic	Quiet time for modulator
PbTtr	int64	r/w	dynam-ic	The Ttr value in t_Bit
PbTtrTypical	int64	r	dynam-ic	Average response time on bus
PbWatchdog	int64	r/w	dynam-ic	Watchdog
PbGapFactor	int	r/w	dynam-ic	Gab update factor
PbRetryLimit	int	r/w	dynam-ic	Maximum number of retries
IsochronousMode	bool	r/w	dynam-ic	True if constant bus cycle time is enabled.
PbAdditionalPassivDeviceForIso-chronousMode	int	r/w	dynam-ic	Number of additional OPs/PGs/TDs etc. that are not configured in this network view.
PbTotalPassivDeviceForIsochro-nousMode	int	r	dynam-ic	Sum of configured and unconfigured devices, such as OPs/PGs/TDs etc.
DpCycleMinTimeAutoCalculation	bool	r/w	dynam-ic	True if automatic calculation and setting of short-est DP cycle time is enabled.
DpCycleTime	double	r/w	dynam-ic	The DP cycle time.

Attribute	Data type	Writa-ble	Access	Description
IsochronousTiToAutoCalculation	bool	r/w	dynam-ic	True if automatic calculation and setting of values of IsochronousTi and IsochronousTo.
IsochronousTi	double	r/w	dynam-ic	Time Ti (read in process values)
IsochronousTo	double	r/w	dynam-ic	Time To (output process values)

### Attributes of subnets of type PROFIBUS Integrated

Attribute	Data type	Writa-ble	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r/w	dynam-ic	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	dynam-ic	Enabled constant bus cycle time.
DpCycleMinTimeAutoCalculation	bool	r/w	dynam-ic	True if automatic calculation and setting of shortest DP cycle time is enabled.
DpCycleTime	double	r/w	dynam-ic	The DP cycle time.
IsochronousTiToAutoCalculation	bool	r/w	dynam-ic	True if automatic calculation and setting of values of IsochronousTi and IsochronousTo.
IsochronousTi	double	r/w	dynam-ic	Time Ti (read in process values)
IsochronousTo	double	r/w	dynam-ic	Time To (output process values)

### Program code

Modify the following program code to get or set the attributes of a subnet:

```
Subnet subnet = ...;

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.



## 7.14.6 Deleting a global subnet

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### Program code

Modify the following program code to delete a a global subnet within a project.:

```
Project project = ...;
SubnetComposition subnets = projects.Subnets;

// delete subnet
Subnet subnetToDelete = ...;
subnetToDelete.Delete();
```

## 7.14.7 Enumerate all participants of a subnet

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### 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
}
```

### 7.14.8 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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
}
```

### 7.14.9 Accessing nodes

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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;
```

## 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.

### 7.14.10 Accessing attributes of a node

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## 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	Writa- ble	Access	Description
Name	string	r		Name of the node.
Nodeld	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.
Address	string	r/w	dynam- ic	Additional attribute for AS-i slaves.

### Attributes of a node of type Ethernet

Attributes	Data type	Writa- ble	Access	Description
Name	string	r		Name of the node.
Nodeld	string	r		ID of the node.
NodeType	NetType	r/w or r		A node gets his type from the subnet.
UseIsoProtocol	bool	r/w	dynam- ic	
MacAddress	string	r/w	dynam- ic	e.g. 01-80-C2-00-00-00
UseIpProtocol	bool	r/w	dynam- ic	This value can be read even if it noct visible at the corresponding TIA UI control.
IpProtocolSelection	enum	r/w	dynam- ic	
Address	string	r/w	dynam- ic	only IPv4 and no IPv6 is supported
SubnetMask	string	r/w	dynam- ic	
UseRouter	bool	r/w	dynam- ic	
RouterAddress	string	r/w	dynam- ic	
DhcpClientId	string	r/w	dynam- ic	
PnDeviceNameSetDirectly	bool	r/w	dynam- ic	PROFINET device name is set directly at the device. Not available for every device.
PnDeviceNameAutoGeneration	bool	r/w	dynam- ic	PROFINET device name is created automatically.

Attributes	Data type	Writa-ble	Access	Description
PnDeviceName	string	r/w	dynam-ic	Unique device name in subnet.
PnDeviceNameConverted	string	r	dynam-ic	Device name converted for system internal using.

### Attributs of a node of type MPI

Attribut	Data type	Writa-ble	Access	Description
Name	string	r		Name of the node.
Nodeld	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.
Address	string	r/w	dynam-ic	

### Attributs of a node of type PC internal

Attribut	Data type	Writa-ble	Access	Description
Name	string	r		Name of the node.
Nodeld	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.

### Attributs of a node of type PROFIBUS

Attribut	Data type	Writa-ble	Access	Description
Name	string	r		Name of the node.
Nodeld	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.
Address	string	r/w	dynam-ic	

### Attributs of a node of type PROFIBUS integrated

Attribut	Data type	Writa-ble	Access	Description
Name	string	r		Name of the node.
Nodeld	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.
Address	string	r	dynam-ic	

**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 = 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).
IpProtocolSelection.OtherPath	IP suite set via other methods, for example PST tool.
IpProtocolSelection.VialoController	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.Ptp	Net type is PTP.

Value	Description
NetType.Wan	Net type is Wide Area Network (WAN).
NetType.Unknown	Net type is Unknow.

### 7.14.11 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 74)
- A project is open.  
See Opening a project (Page 97)

#### Program code

Modify the following program code to assign a node (device, interface) to a network:

```
Node node = ...;  
Subnet subnet = ...;  
node.ConnectToSubnet (subnet);
```

### 7.14.12 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 74)
- A project is open.  
See Opening a project (Page 97)

#### Program code

Modify the following program code to disconnect a node (device, interface) from a network:

```
Node node = ...;  
node.DisconnectFromSubnet ();
```

### 7.14.13 Creating an io system

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### Application

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 acquired 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 interface = ...;
IoSystem ioSystem = null;

// Interface is configured as io controller
if((interface.InterfaceOperatingMode & InterfaceOperatingModes.IoController) != 0)
{
    IoControllerComposition ioControllers = interface.IoControllers;
    IoController ioController = ioControllers.First();
    if(ioController != null)
    {
        ioSystem = ioController.CreateIoSystem("io system");
    }
}
```



## 7.14.14 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 74)
- A project is open.  
See Opening a project (Page 97)

### 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;
}
```

### Program code: Subnet of an io system

Modify the following program code to navigat to the subnet the io system is assigned to:

```
Subnet subnet = ioSystem.Subnet;
```

## 7.14.15 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 74)
- A project is open.  
See Opening a project (Page 97)

### 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 194).

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();
IoController ioController = ioConnector.GetIoController();
```

### 7.14.16 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 74)
- A project is open. See Opening a project (Page 97)

#### 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
}
```

## 7.14.17 Get an IO Controller

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### 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
}
```

### 7.14.18 Get an IO Connector

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### Application

An IoConnector does not provide any modelled attributes or actions.

#### Program code

Modify the following program code to get the io connector:

```
NetworkInterface itf = ...
foreach (IoConnector ioConnector in itf.IoConnectors)
{
    // work with the IoConnector
}
```

### 7.14.19 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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 194).

## Program code

Modify the following program code:

```
IoSystem ioSystem = ...;  
IoConnector ioConnector = ...;  
  
ioConnector.DisconnectFromIoSystem();
```

## 7.14.20 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 74)
- A project is open.  
See Opening a project (Page 97)

### Application

A DP Mastersystem 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 DP Master and the DP Slaves which are assigned to this DP Mastersystem.

### Attributes of a dp mastersystem

Attribute	Data type	Writa-ble	Access	Description
Name	string	r/w		
Number	int	r/w		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;
```

**7.14.21 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 74)
- A project is open. See Opening a project (Page 97)

**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	Writ-able	Access	Description
MultipleUseIoSystem	bool	r/w	dynam-ic	
Name	string	r/w		
Number	int	r/w		The attribute Number accepts values that cannot be set via UI. In this case the compile will fail.
UseIoSystemNameAsDeviceNameExtension	bool	r/w	dynam-ic	If MultipleUseIoSystem is set to TRUE, UseIoSystemNameAsDeviceNameExtension will be set to FALSE and write access is not possible.
MaxNumberIWlanLinksPerSegment	int	r/w	dynam-ic	

### 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);
```

## 7.14.22 Deleting a dp mastersystem

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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();
```

### 7.14.23 Deleting a profinet io system

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

#### Program code

Modify the following program code to delete a profinet io system:

```
IoController ioController = ...;
IoSystem ioSystem = ioController.IoSystem;
ioSystem.Delete();
```

### 7.14.24 Creating a dp master system

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

#### Application

A DP master system is created by calling the action `CreateIoSystem(string nameOfIoSystem)` on an object of the type `IoController`. The io controller is acquired 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:

```
using System.Linq;
...
NetworkInterface interface = ...;
IoSystem dpMasterSystem = null;

// Interface is configured as master or as master and slave
if((interface.InterfaceOperatingMode & InterfaceOperatingModes.IoController) != 0)
{
    IoControllerComposition ioControllers = interface.IoControllers;
    IoController ioController = ioControllers.First();
    if(ioController != null)
    {
        dpMasterSystem = ioController.CreateIoSystem("dp master system");
    }
}
```

## 7.14.25 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 74)
- A project is open.  
See Opening a project (Page 97)

### 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 exceptions 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);
```

**7.14.26 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 modify 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	
AlternativePartner-Ports	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

## 7.14.27 Accessing the attributes of a port

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

## 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;
```

### 7.14.28 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### 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
}
```

### 7.14.29 Enumerate assigned io connectors

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

#### 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
}
```

### 7.14.30 Connecting a dp io connector to a dp mastersystem

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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 194).

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();
```



## 7.15 Functions on devices

### 7.15.1 Mandatory attributes of devices

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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);
}
```

## 7.15.2 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 74)
- A project is open.  
See Opening a project (Page 97)

### 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 version existing in the system. Be care
<OrderNumber>// <AdditionalTypeIdentifier>	OrderNumber:6AV2 124-2DC01-0AX0//Landscape	The additional type identifier might be necessary in case that <code>OrderNumber</code> and <code>FirmwareVersion</code> 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	<code>GsdName</code> is the name of the GSD or GSDML in up-percase letters.
<GsdName>/<GsdType>/<GsdId>	GSD:SIEM8139.GSD/M/4	<code>GsdType</code> is one of the following: <ul style="list-style-type: none"> <li>• D: Device</li> <li>• R: Rack</li> <li>• DAP: HeadModule</li> <li>• M: Module</li> <li>• SM: Submodule</li> </ul> <code>GsdId</code> is a identifier for the type.

## System

This is the identifier for objects, which cannot be determined using `OrderNumber` or `GSD`.

Format of type identifier	Example	Specifics
<code>&lt;SystemTypeIdentifier&gt;</code>	<code>System:Device.S7300</code>	<code>SystemTypeIdentifier</code> is the primary identifier of an object.
<code>&lt;SystemTypeIdentifier&gt;/ &lt;AdditionalTypeIdentifier&gt;</code>	<code>GSD:SIEM8139.GSD/ M/4</code>	<p><code>AdditionalTypeIdentifier</code> might be necessary in case the <code>SystemTypeIdentifier</code> is not unique. The prefix for certain object types are:</p> <ul style="list-style-type: none"> <li>• Connection.</li> <li>• Subnet.</li> <li>• Device.</li> <li>• Rack.</li> </ul>

## Program code

Modify the following program code to get the type identifier for user manageable and separately creatable objects for GSD:

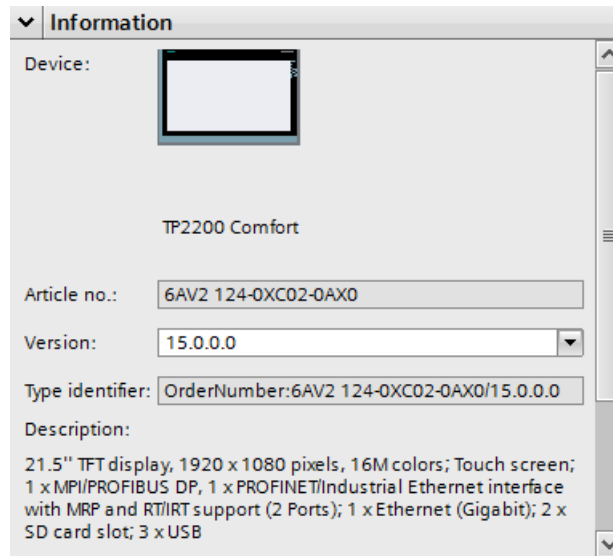
```
HardwareObject hardwareObject = ...;
string typeIdentifier = hardwareObject.TypeIdentifier;
```

## 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"



### 7.15.3 Creating a device

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

#### Application

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(DeviceItemId, DeviceItemName, DeviceName)`
- Create only the device  
`Device Create(DeviceTypeId, DeviceName)`

Name	Type	Description
DeviceItemId	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: Type identifier (Page 210)

### 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");
```

## 7.15.4 Enumerating devices

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### Application: Enumerating devices

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 64).

---

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.

Modify the following program code:

```
//Enumerate 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
    }
}
```

### Programm 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
```

## 7.15.5 Accessing devices

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

## Application

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		TRUE in case of a GSD device or a GSDML device
Name	string	read/write		
TypeIdentifier	string	read		

### 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

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);
}
```

### 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

Modify the following program code to get the attributes:

```
Device device = ...;
GsdDevice gsdDevice = ...;
string gsdId = gsdDevice.GsdId;
string gsdName = gsdDevice.GsdName;
string gsdType = gsdDevice.GsdType;
bool isProfibus = gsdDevice.IsProfibus;
bool isProfinet = gsdDevice.IsProfinet;
```

## 7.15.6 Deleting a device

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

## **Program code**

Modify the following program code to delete a device:

```
Project project = ...;  
Device deviceToDelete = project.UngroupedDevices.Devices.Find(".....");  
  
// delete device  
deviceToDelete.Delete();
```

## 7.16 Functions on device items

### 7.16.1 Mandatory attributes of device items

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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		
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 <code>NetworkInterface</code>
InterfaceType	NetType	readwrite	dynamic	For device items that provides the feature <code>NetworkInterface</code>
IsBuiltIn	bool	read		FALSE for objects creatable by the user
IsGsd	bool	read		TRUE, if the device description is installed via GSD/GSDML
IsPlugged	bool	read		TRUE for devices that are plugged
Label	string	read	dynamic	For device items that provides the feature <code>NetworkPort</code> or <code>NetworkInterface</code> . If the interface or port has no label <code>Label</code> will be <code>String.Empty</code> .
LocationIdentifier	string	read/write	dynamic	
Name	string	read/write		sometimes only read access
OrderNumber	string	read/write	dynamic	sometimes only read access
PlantDesignation	string	read/write	dynamic	
PositionNumber	int	read		
TypeIdentifier	string	read		
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).

**Device item classification**

Value	Description
DeviceItemClassifications.None	No classification.
DeviceItemClassifications.CPU	The device item is a CPU
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.");
```

**7.16.2 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 74)
- A project is open.  
See Opening a project (Page 97)

## 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 successful 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
<code>typeIdentifier</code>	string	type identifier of the created device item
<code>name</code>	string	name of created device item
<code>positionNumber</code>	int	position number of the created device item

## Program code

Modify the following program code to plug a device item into an existing 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 FindContainerTypes method

FindContainerTypes method returns the container types of a given module. The module is specified using 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 least one part. Ex. `System:Rack.ET200SP`

Name of the parameter	Type	Description
<code>partialTypeIdentifier</code>	string	partial type identifier 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 properties.

Name of the property	Type	Description
<code>PositionNumber</code>	int	The position number of the free slot
<code>Label</code>	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.

```
IHardwareObject hardwareObject = ...;
IList<PlugLocation> result = hardwareObject.GetPlugLocations();
foreach (PlugLocation item in result)
{
    Console.WriteLine("{0} - {1}", item.PositionNumber, item.Label);
}
```

### 7.16.3 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 74)
- A project is open.  
See Opening a project (Page 97)

#### Application

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);
}
```

## 7.16.4 Copying a device item

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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
<code>deviceItem</code>	<code>DeviceItem</code>	Device item to copy
<code>positionNumber</code>	<code>int</code>	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);
}
```

### 7.16.5 Deleting a device item

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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();
```

### 7.16.6 Enumerate device items

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## 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 64).

---

## 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
    }
}
```

**7.16.7 Accessing device items****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

**Application: Accessing device items**

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		TRUE in case of a GSD device or a GSDML device
IsPlugged	bool	read		
IsProfibus	bool	read		
IsProfinet	bool	read		
Name	string	read/write		

Name	Data type	Writeable	Access	Description
OrderNumber	string	read	dynamic	Only for head modules
PositionNumber	bool	read		
TypeIdentifier	string	read		

### 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

Modify the following program code to get the prm data:

```
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 64)

## 7.16.8 Accessing device item as interface

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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 be used as `IoDevice` (Slave) or `IoController` (Master) by accessing the `NetworkInterface` feature (a specific Service of the device item).

The properties of the interface are accessed using enum `InterfaceOperatingModes`.

Value	Description
<code>InterfaceOperatingModes.None</code>	Default
<code>InterfaceOperatingModes.IoDevice</code>	Interface operation mode "IoDevice" (Slave).
<code>InterfaceOperatingModes.IoController</code>	Interface operation mode "IoController" (Master).
<code>InterfaceOperatingModes.IoDevice</code> or <code>InterfaceOperatingModes.IoController</code>	Interface operation mode both of the above.

## Program code: Accessing the network interface feature

Modify the following program code to get the network interface feature

```

NetworkInterface itf =
((IEngineeringServiceProvider)deviceItem).GetService<NetworkInterface>();
if (itf != null)
{
... // work with the interface
}

//Accessing nodes and operating mode
NodeComposition nodes = itf.Nodes;
InterfaceOperationModes mode = itf.InterfaceOperatingMode;

//Accessing the type of interface
NetType itfType = itf.InterfaceType;

//Modifying the operating mode and interface type
itf.InterfaceOperatingMode = InterfaceOperationModes.IoDevice;
itf.InterfaceType = NetType.Profibus

//Accessing the ports linked to an interface.
NetworkPortAssociation nodes = itf.Ports;

```

## 7.16.9 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- For writing access, the PLC is offline.

### Application

You can use the TIA Portal Openness API interface to get or set attributes for IRT and 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
IsynchronousTiToAutoCalculation	BOOL	r/w	Dynamic attribute
IsynchronousTi	DOUBLE	r/w	Dynamic attribute
IsynchronousTo	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 system.

Attribute name	Data type	Writeable	Access
IsynchronousMode	BOOL	r/w	Dynamic attribute
IsynchronousTiToCalculationMode	IsynchronousTiToCalculationMode	r/w	Dynamic attribute
IsynchronousTi	DOUBLE	r/w	Dynamic attribute
IsynchronousTo	DOUBLE	r/w	Dynamic attribute

The following ENUM values are provided for the attribute  
IsynchronousTiToCalculationMode:

Value	Description
IsynchronousTiToCalculationMode.None	
IsynchronousTiToCalculationMode.FromOB	Ti/To values of the OB (configured at the IoSystem) are used.
IsynchronousTiToCalculationMode.FromSubnet	This value is not used by PROFINET interfaces.
IsynchronousTiToCalculationMode.AutomaticMinimum	Ti/To values are calculated automatically for the IO Device.
IsynchronousTiToCalculationMode.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);
```

## 7.16.10 Accessing attributes of IoController

### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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 Synchronization 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);
```

## 7.16.11 Accessing attributes of IoConnector

### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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 Openness.

There are four types of attributes such as update time attributes, watchdog time attributes, 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 97)

## 7.16.12 Accessing address controller

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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)  
{  
    ...  
}
```

### 7.16.13 Accessing addresses

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### Application

Address objects are acquired 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		
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 not 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");
```

## 7.16.14 Accessing hardware identifiers

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### Application

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 HwIdentifier
    }
```

**Attributes of a hardware identifier**

```
HwIdentifierControllerAssociation controllers = hwIdentifier.HwIdentifierControllers;
Int64 Identifier = hwIdentifier.Identifier;
```

**7.16.15 Accessing hardware identifier controller****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

**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;
```

## 7.16.16 Accessing channels of device items

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### Application

A channel is represented by the Channel class. Channels are acquired 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		
Type	ChannelType	read		
Number	Int32	read		
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);
```

## 7.17 Functions for accessing the data of an HMI device

### 7.17.1 Screens

#### 7.17.1.1 Creating user-defined screen folders

##### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

##### 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");
}
```

#### 7.17.1.2 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 74)
- A project is open. See Opening a project (Page 97)

##### 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();
    }
}
```

### 7.17.1.3 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- 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();
    }
}
```



### 7.17.1.4 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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();
    }
}
```

## 7.17.2 Cycles

### 7.17.2.1 Deleting a cycle

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)
- 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();
}
```

## 7.17.3 Text lists

### 7.17.3.1 Deleting a text list

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)
- 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();
}
```

## 7.17.4 Graphic lists

### 7.17.4.1 Deleting a graphic list

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)
- 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();
}
```

## 7.17.5 Connections

### 7.17.5.1 Deleting a connection

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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();
}
```

## 7.17.6 Tag table

### 7.17.6.1 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 74)
- A project is open.  
See Opening a project (Page 97)

## 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");
}
```

### 7.17.6.2 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

## 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
    }
}
```

### 7.17.6.3 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

## 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();
}
```

### 7.17.6.4 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 \(Page 74\)](#)
- A project is open. See [Opening a project \(Page 97\)](#)
- 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();
}
```

## 7.17.7 VB scripts

### 7.17.7.1 Creating user-defined script folders

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### 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 = vbScriptFolders.Create("mySubfolder");
}
```

### 7.17.7.2 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- The project contains an HMI device.

## 7.17 Functions for accessing the data of 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();
    }
}
```

### 7.17.8 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 74)
- A project is open.  
See Opening a project (Page 97)

### 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();
```



## 7.18 Functions for accessing the data of a PLC device

### 7.18.1 Determining the status of a PLC

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application. See [Opening a project \(Page 97\)](#)

#### 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 accessible
- 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;
            }
        }
    }
}
```

## 7.18.2 Accessing parameters of an online connection

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### 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.

**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;
}
```

---

## 7.18 Functions for accessing the data of a PLC device

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}", gatewayAddress.Name);
            }
        }
    }
}
```

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();
}
```

### 7.18.3 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 74)
- A project is open.  
See Opening a project (Page 97)

#### Application

You can use the RHOonlineProvider service to set online either to primary PLC or backup PLCs of R/H system.

#### Program code: Accessing RHOonlineProvider service from a device

Modify the following code to access RHOonlineProvider:

```
Device device = project.Devices.Find("S7-1500R/H-System_1");  
RHOonlineProvider rhOnlineProvider = device.GetService<RHOonlineProvider>();
```

#### 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 RHOonlineProvider. For more information about how to set connection, Refer Accessing parameters of an online connection (Page 258)

Modify the following program code to set a connection mode and access a PC interface by name:

```
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 \(Page 257\)](#)

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 \(Page 258\)](#)

[Determining the status of a PLC \(Page 257\)](#)

[Connecting to the TIA Portal \(Page 74\)](#)

[Opening a project \(Page 97\)](#)

## 7.18.4 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### 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 SoftwareContainer and its software property are described in [Access software target. \(Page 111\)](#)

### 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 \(Page 111\)](#)
- [Connecting to the TIA Portal \(Page 74\)](#)
- [Opening a project \(Page 97\)](#)



## 7.18.5 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 74)
- A project is open.  
See Opening a project (Page 97)

### Application

You can use the TIA Portal Openness application to download the primary and backup PLCs of R/H system. You should be able to download both hardware and software components of the system. (Refer Downloading hardware and software components to PLC device (Page 271))

### 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.

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(IConfiguration
configuration, DownloadConfigurationDelegate
preDownloadConfigurationDelegate, DownloadConfigurationDelegate
postDownloadConfigurationDelegate, DownloadOptions downloadOptions);
```

Modify the following program code to download to the backup CPU by invoking `RHDownloadProvider.DownloadToBackup`:

```
DownloadResult DownloadToBackup(IConfiguration configuration,
DownloadConfigurationDelegate preDownloadConfigurationDelegate,
DownloadConfigurationDelegate postDownloadConfigurationDelegate, DownloadOptions
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 271)

Parameter name	Type	Description
configuration	Siemens.Engineering.Connection.IConfiguration	Connection configuration to a device.
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 271), the following data type are added to support RHDDownload.

Configuration	Data Type	Action	Description
DownloadSelection-Configuration	StopHSystem	Set CurrentSelection:StopH-SystemSelections. Available enum values: <ul style="list-style-type: none"> <li>• NoAction (No Action)</li> <li>• StopHSystem (Stop R/H-System)</li> </ul>	The modules are stopped for downloading to device
	StopHSystemOrModule	Set CurrentSelection:StopH-SystemOrModuleSelections. Available enum values: <ul style="list-style-type: none"> <li>• NoAction (No action)</li> <li>• StopHSystem (Stop R/H-System)</li> <li>• StopModule (Stop module)</li> </ul>	The modules are stopped for downloading to device
	StartBackupModules	Set CurrentSelection:Start-BackupModulesSelections. Available enum values: <ul style="list-style-type: none"> <li>• NoAction (No action)</li> <li>• SwitchToPrimaryCpu (Change to Primary )</li> <li>• StartModule (Start module)</li> </ul>	Start modules after downloading to device
	SwitchBackupToPrimary	Set CurrentSelection:Switch-BackupToPrimarySelections. Available enum values: <ul style="list-style-type: none"> <li>• NoAction (No action)</li> <li>• SwitchToPrimaryCpu (Change to Primary)</li> </ul>	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 overwriteSystemData = downloadConfiguration as OverwriteSystemData;
    if (overwriteSystemData != null)
    {
        overwriteSystemData.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 \(Page 74\)](#)

[Opening a project \(Page 97\)](#)

[Downloading hardware and software components to PLC device \(Page 271\)](#)

## 7.18.6 Functions for downloading data to PLC device

### 7.18.6.1 Downloading hardware and software components to PLC device

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- You have opened a project with your TIA Portal Openness application. See Opening a project (Page 97)

#### Application

Openness user is able 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)
{
    ...
}
```

#### Parameters of download method

In order to download to a PLC device, the user calls 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.
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.

## 7.18 Functions for accessing the data of a PLC device

- Openness download is supported only if the configurations are handled properly by the user. If the configuration is invalid, then `EngineeringTargetException` is thrown and download process is aborted. The F-activated PLCs are not supported for download operation.
- Since compilation is a part of download, it is recommended to compile before the download operation to analyze the compile results.
- Openness supports only full download option.

### Parameter 1: IConfiguration

The user should provide `IConfiguration` object as first parameter for the `Download` method. It is 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 258)` 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];
...
```

### Parameter 2 and 3: DownloadConfigurationDelegate

Openness user need 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 282)`. Certain configurations will only contain an information, therefore the user action will not be required.

The possible download configuration types are listed below.



Configuration name	Description and properties
DownloadConfiguration	<ul style="list-style-type: none"> <li>• Base class for all the configurations.</li> <li>• Contains single property DownloadConfiguration.Message : string (read only property contains the configuration message)</li> </ul>
DownloadSelectionConfiguration	<ul style="list-style-type: none"> <li>• Base class for all configuration that can be selected.</li> <li>• Does not contain additional properties. A selection must be provided in all child classes derived from it.</li> </ul>
DownloadCheckConfiguration	<ul style="list-style-type: none"> <li>• Base class for all configuration that can be checked and unchecked.</li> <li>• Contains single property DownloadCheckConfiguration.Checked: bool&lt;string&gt; (read/write property identifies whether the configuration is checked or unchecked)</li> </ul>
DownloadPasswordConfiguration	<ul style="list-style-type: none"> <li>• Base class for all configuration that required a password for download.</li> <li>• Contains a single method to set password. DownloadPasswordConfiguration.SetPassword (password: SecureString) : void</li> </ul>

The datatype of the configurations are given below.



## 7.18 Functions for accessing the data of a PLC device

Configuration	Data Type	Description and Action
DownloadSelection-Configuration	StartModules	Set CurrentSelection:StopModulesSelections. Available enum values are NoAction (No action) StopAll (Stop all) These modules are stopped for downloading to a device.
	StopModules	Set CurrentSelection:StartModulesSelections. Available enum values: NoAction (No action) StartModule (Start module) These modules are started after the download operation.
	AllBlocksDownload	Set CurrentSelection:AllBlocksDownloadSelections. Available enum value is DownloadAllBlocks (Download all blocks to the device) Downloads software to device
	OverwriteSystemData	Set CurrentSelection:OverwriteSystemDataSelections. Available enum values are NoAction (No action) Overwrite (Download to device) Deletes and replaces the existing system data in target location.
	ConsistentBlocksDownload	Set CurrentSelection:ConsistentBlocksDownloadSelections. Available enum value is ConsistentDownload (Consistent download) Downloads the software to device.
	AlarmTextLibrariesDownload	Set CurrentSelection:AlarmTextLibrariesDownloadSelections. Available enum values are ConsistentDownload (Consistent download) NoAction (No action) Downloads 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 loading operation of the device.
	ResetModule	Set CurrentSelection:ResetModuleSelections. Available enum values are

## 7.18 Functions for accessing the data of a PLC device

Configuration	Data Type	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) LoadSelected (Load selected) 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.
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 the objects 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.

**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.

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.

**PreDownloadDelegate implementation example:**

```
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
}
```

**PostDownloadDelegate implementation example:**

```
private static void PostConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    StartModules startModules = downloadConfiguration as StartModules;
    if (startModules != null)
    {
        startModules.CurrentSelection = StartModulesSelections.StartModule; //
Sets PLC in "Run" mode
    }
}
```

**Parameter 4: DownloadOptions**

The user must specify the download options through DownloadOptions flagged enum. This parameter will determine the type of download to be performed such as Hardware, Software or Hardware and Software.

```
[Flags]
public enum DownloadOptions {
    None = 0, // Download nothing
    Hardware, // Download hardware only
    Software // Download software only
}
```

If the user wants to download both Software and Hardware to the device, then pass `DownloadOptions.Hardware | DownloadOptions.Software` as the 4th parameter of the Download method.

## DownloadResult

The DownloadResult returned by the Download action provides feedback on the state of the objects that were downloaded.

### Download invocation example

```
[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);
    }
}
}
```



Modify the following code to download PLC by calling applying configuration:

```
private static void DownloadNCU(Device ncu, ConfigurationTargetInterface
configurationTargetInterface)
{
    DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
    DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
    DownloadProvider downloadProvider = null;
    foreach (var item in ncu.DeviceItems[0].DeviceItems)
    {
        downloadProvider = item.GetService<DownloadProvider>();
        if (downloadProvider != null)
        {
            break;
        }
    }
    downloadProvider.Configuration.ApplyConfiguration(configurationTargetInterface);
    IConfiguration targetConfiguration = configurationTargetInterface;
    downloadProvider.Download(targetConfiguration, preDownloadDelegate,
postDownloadDelegate, DownloadOptions.Hardware | DownloadOptions.Software);
}
```

## 7.18.6.2 Running and stopping PLC

### Requirements

- The Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- 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 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;
    }
}
```

### 7.18.6.3 Supporting callbacks

#### Requirements

- The Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### 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.

### 7.18.6.4 Protecting PLC through password

#### Requirements

- The Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)
- 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 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.

### 7.18.6.5 Handling PLC block binding passwords

#### Requirements

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- PLC is offline.

#### Application

TIA Openness supports the data binding of passwords for customer applications. TIA 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);
    }
}
```

### 7.18.7 Uploading hardware, software and files to PLC device

#### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application.  
See [Opening a project \(Page 97\)](#)

#### Application

Openness user is able to upload station into a project through `StationUploadProvider` (accessed from a given project). An upload into a `DeviceGroup` is not supported. If a project is used to execute an upload, an instance of `StationUploadProvider` will be returned on `GetService` call, else the service will return null.

#### 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, user calls StationUpload method of StationUploadProvider. The Upload method has ConfigurationAddress and UploadConfigurationDelegate 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

The user 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

The user 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

## 7.18 Functions for accessing the data of a PLC device

```

...
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**

Openness user needs 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 282). 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	<ul style="list-style-type: none"> <li>• Base class for all the configurations. It contains information in the Message attribute</li> <li>• Contains single property UploadConfiguration.Message : string (read only property contains the configuration message)</li> </ul>
UploadPasswordConfiguration	<ul style="list-style-type: none"> <li>• Derived from UploadConfiguration</li> <li>• Base class for all configuration that required a password for upload.</li> <li>• Contains a single method to set password. UploadPasswordConfiguration.SetPassword (password: SecureString) : void - Set password</li> </ul>
UploadSelectionConfiguration	<ul style="list-style-type: none"> <li>• Derived class of UploadConfiguration</li> <li>• Does not contain additional properties</li> </ul>



The datatype of the configurations are given below:

Configuration	Data Type	Description and Action
UploadPasswordConfiguration	ModuleReadAccessPassword	Set password via <code>SetPassword(password:SecureString)</code> method. Enter a password to gain read access to the module.
	PasswordReadAccess	Set password via <code>SetPassword(password:SecureString)</code> method. Enter a password for SW Upload in classic PLC's to gain read access to the module.
UploadSelectionConfiguration	UploadMissingProducts	Set <code>CurrentSelection:UploadMissingProductsSelections</code> Available enum values: <code>TryUpload</code> (Consistent upload) <code>NoAction</code> (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.

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**

The user cannot specify the Upload options. This Upload options are known as: "Hardware", "Software", "Hardware and Software" and "Hardware, Software and Files".

**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 upload
State	UploadResultState with possibly values: Success, Information, Warning and Error

Attributes	Description
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);
    }
}
```

## 7.18.8 Accessing fingerprints

### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- You have opened a project with your TIA Portal Openness application.  
See Opening a project (Page 97)

### Application

You can use the TIA Portal Openness API to detect changes inside of blocks or UDT. You can achieve this by comparing the fingerprints of the object. A fingerprint instance contains an `FingerprintId` which defines the kind of fingerprint 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.

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
TextualInterface	Exists when the block has a textual interface

## Program code

You need to use the `FingerprintProvider` service to retrieve the fingerprints of an object. It is available for blocks (FB, FC, OB, DB), and UDTs, but not for tag tables. The `FingerprintProvider` calculates and returns all available fingerprints of an object with each `GetFingerprints` call. In order to ensure correct fingerprints, the block or UDT needs to be consistent before calling fingerprints. Otherwise, an `RecoverableException` is thrown. When a fingerprint is still invalid after its calculation, an `RecoverableException` is thrown.

```
PlcBlock block = ...;
FingerprintProvider provider = block.GetService<FingerprintProvider>();
IList<Fingerprint> fingerprints = provider.GetFingerprints();
foreach(var fingerprint in fingerprints)
{
    string fpValue = fingerprint.Value;
    FingerprintId fpId = fingerprint.Id;
}
```

## See also

[Connecting to the TIA Portal \(Page 74\)](#)

[Opening a project \(Page 97\)](#)

## 7.18.9 Comparing PLC software

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See [Connecting to the TIA Portal \(Page 74\)](#)
- You have opened a project with your TIA Portal Openness application. See [Opening a project \(Page 97\)](#)

### 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
- Comparison of 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: <ul style="list-style-type: none"> <li>• FolderContentsDifferent: Content of the compared folders differs.</li> <li>• FolderContentsIdentical: Content of the compared folders is identical.</li> <li>• ObjectsDifferent: Content of the compared objects differs.</li> <li>• ObjectsIdentical: Content of the compared objects is identical.</li> <li>• LeftMissing: The object is not contained in the object from which the comparison was started.</li> <li>• RightMissing: The object is not contained in the object which is being compared.</li> </ul>
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);
    }
}
```

### 7.18.10 Comparing PLC hardware

#### Requirements

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- You have opened a project with your TIA Portal Openness application. See Opening a project (Page 97)

#### Application

You can use the TIA Openness API 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 comparison result: <ul style="list-style-type: none"> <li>• FolderContainsDifferencesOwnStateDifferent: Folder contents have one or more differences, folder's own state is different</li> <li>• FolderContentEqualOwnStateDifferent: Folder content is the same, folder's own state is different.</li> </ul>
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.Engineering.EngineeringTargetInvocationExceptions.



## Program

Modify the following program code to output the comparison 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 74)

Opening a project (Page 97)

## 7.18.11 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 74)
- A project is open.  
See Opening a project (Page 97)
- All devices are enumerated.  
See Accessing device items (Page 230).

### 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();
            }
        }
    }
}
```

## 7.18.12 Blocks

### 7.18.12.1 Querying the "Program blocks" group

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- 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;
}
```

### 7.18.12.2 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

### 7.18 Functions for accessing the data of a PLC device

#### 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
        }
    }
}
```

#### 7.18.12.3 Enumerating system subgroups

##### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

**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 313)

**7.18.12.4 Enumerating user-defined block groups****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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");
}
```

#### 7.18.12.5 Enumerating all blocks

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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");
}
```

#### 7.18.12.6 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 74)
- A project is open.  
See Opening a project (Page 97)

## 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 56) 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 417)

### 7.18.12.7 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 74)
- A project is open.  
See Opening a project (Page 97)
- PLC is not online.

#### Application

You can set or remove the password protection of a block via the ProtectionProvider class and the ProtectionProvider service. The service ProtectionProvider is accessible on blocks which fulfill the following conditions:

- block is know-how protectable
- block is a code block or a global DB
- 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.

#### Program code: Performing know-how protection related operations

Modify the following program code:

```
PlcBlock block = ...;

ProtectionProvider protectionProvider = block.GetService<ProtectionProvider>();
if (protectionProvider != null)
{
    ... // perform know-how protection related operations here
}
```

## Protect a block

Use the `Protect()` method to set the password to protect the programming block with.

```
void Protect(SecureString password)
```

Errors will occur in case

- of an attempt to protect an already protected block: An `EngineeringTargetInvocationException` will be thrown with the message "You can't protect an already protected object".
- of an attempt to protect with an empty string as password: An `EngineeringTargetInvocationException` will be thrown with the message "Password was not specified".
- of an attempt to protect a failsafe block when the failsafe-program is password protected: An `EngineeringTargetInvocationException` will be thrown.
- of 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 programming block is protected with.

```
void Unprotect(SecureString password)
```

Errors will occur in case

- of an attempt to unprotect an already unprotected block: An `EngineeringTargetInvocationException` will be thrown with the message "You can't unprotect an object without protection".
- of an attempt to unprotect with wrong password: An `EngineeringTargetInvocationException` will be thrown with the message "The used password was refused".
- of an attempt to protect with an empty string as password: An `EngineeringTargetInvocationException` will be thrown with the message "Password was not specified".

## Check for invalid characters

Because you can use any characters including backspace, tab etc. to protect a block with the `Protect()` method, it could be impossible to remove the protection within the TIA Portal. Since the passwords are submitted as `SecureString`, you have to check for yourself if the provided password has illegal characters. With the `GetInvalidPasswordCharacters()` method you can retrieve a list of 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 case

- of an attempt to unprotect an already unprotected block: An `EngineeringTargetInvocationException` will be thrown with the message "You can't unprotect an object without protection".
- of an attempt to unprotect with wrong password: An `EngineeringTargetInvocationException` will be thrown with the message "The used password was refused".
- of an attempt to protect with an empty string as password: An `EngineeringTargetInvocationException` will be thrown with the message "Password was not specified".

### 7.18.12.8 Deleting block

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- 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 417)

### 7.18.12.9 Creating group for blocks

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## 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 417)

### 7.18.12.10 Deleting group for blocks

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- 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 \(Page 417\)](#)

### 7.18.12.11 Accessing attributes of all blocks

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### Application

User can set the attributes applicable for all blocks using `SetAttribute()` method. The following code examples were given based on the two attributes `AutoNumber` and `Number` (Refer [Exporting blocks \(Page 486\)](#) for all applicable attributes of blocks).

## 7.18 Functions for accessing the data of a PLC device

### Program code:

```
...
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);
...
```

### 7.18.12.12 Creating a ProDiag-FB

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

#### Application

Openness user can use the PLCBlock composition's create action with the following parameters to create ProDiag FB.

1. Name
  2. Auto number flag
  3. Number (ignore in case of "auto number flag" is true)
  4. Programming language
- If the user invoke create action with ProDiag programming language, then a new FB will be created without IDB.
  - If user 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

```
...
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 311)

### 7.18.12.13 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 74)
- A project is open.  
See Opening a project (Page 97)

#### Accessing supervisions of User-FB

Openness user can access the supervisions at FB by using the following code snippet. Every FB has the list of supervisions including Classic and Plus PLCs.

**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");
...
```

**Accessing the attributes of FB block**

Openness user can set AssignedProDiagFB at InstanceDB via the attribute AssignedProDiagFB (Refer Exporting blocks (Page 486)). The user can use `GetAttribute()`, `GetAttributes()` and `SetAttribute()` method for accessing the attributes. The user cannot use `SetAttributes()` method for setting the attributes for more than one attribute. TIA Portal Openness throws an exception for using `SetAttributes()` method.

If the attribute is not supported (in the given block), recoverable user exception is thrown. If there is no assigned ProDiag-Block set, `GetAttribute()` returns an empty string.

**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 \(Page 310\)](#)



### 7.18.12.14 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 74)
- You have opened a project via a TIA Portal Openness application  
See Opening a project (Page 97)

#### 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 74)

Opening a project (Page 97)

### 7.18.12.15 Adding an external file

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- You have opened a project via a TIA Portal Openness application:  
See Opening a project (Page 97)

#### 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");
}
```

### 7.18.12.16 Generate source from block

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- PLC is not online.

#### Application

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:

```
//method declaration
...
PlcExternalSourceSystemGroup.GenerateSource

(IEnumerable<Siemens.Engineering.SW.ExternalSources.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 417)

### 7.18.12.17 Generating blocks from source

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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

Access to groups in the "External source files" folder is not supported.

Existing blocks are overwritten.

An Exception is thrown if an error occurs during the calling. The first 256 characters of each error message are contained in the notification of the Exception. 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();
}
```

### 7.18.12.18 Deleting user data type

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- PLC is not online.

## Program code

Modify the following program code to delete a user type:

```
private static void DeleteUserDataTypes(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 417)

### 7.18.12.19 Deleting an external file

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal see Connecting to the TIA Portal (Page 74)
- You have opened a project via a TIA Portal Openness application: see Opening a project (Page 97)
- 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();
}
```

### 7.18.12.20 Starting the block editor

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- 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 StartPlcTypeEditor(PlcSoftware plcSoftware)
{
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find("my_udt");
    udt.ShowInEditor();
}
```

#### See also

[Importing configuration data \(Page 417\)](#)

## 7.18.13 Technology objects

### 7.18.13.1 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.

You get the code components that have to be adapted for each task.

## Functions

The following functions are available for technology objects:

- Querying the composition of technology objects (Page 323)
- Creating technology object (Page 323)
- Deleting technology object (Page 324)
- Compiling technology object (Page 325)
- Enumerating technology object (Page 326)
- Finding technology object (Page 327)
- Enumerating parameters of technology object (Page 328)
- Finding parameters of technology object (Page 328)
- Reading parameters of technology object (Page 329)
- Writing parameters of technology object (Page 330)

## See also

Standard libraries (Page 40)

Applications (Page 34)

TIA Portal Openness object model (Page 51)

### 7.18.13.2 Overview of technology objects and versions

#### Technology objects

The following table shows the available technology objects in the Public API.

CPU	FW	Technology object	Version of technology object
S7-1200	≥ V4.2	TO_PositioningAxis	V6.0
		TO_CommandTable	
		PID_Compact	V2.3
		PID_3Step	
		PID_Temp	V1.1



CPU	FW	Technology object	Version of technology object
S7-1500	< V2.0	High_Speed_Counter	V3.0
		SSI_Absolute_Encoder	V2.0
	≥ V2.0	TO_SpeedAxis	≥ V3.0
		TO_PositioningAxis	
		TO_ExternalEncoder	
		TO_SynchronousAxis	
		TO_OutputCam	
		TO_CamTrack	
		TO_MeasuringInput	
		TO_Cam (S7-1500T) <sup>1)</sup>	
		TO_Kinematics (S7-1500T)	V4.0
		High_Speed_Counter	≥ V3.0
		SSI_Absolute_Encoder	≥ V2.0
		PID_Compact	≥ V2.3
		PID_3Step	V2.3
		PID_Temp	V1.1
	CONT_C		
	CONT_S		
	TCONT_CP		
TCONT_S			
S7-300/400	Any	CONT_C	V1.1
		CONT_S	
		TCONT_CP	
		TCONT_S	
		TUN_EC <sup>2)</sup>	
		TUN_ES <sup>2)</sup>	
		PID_CP <sup>2)</sup>	V2.0
		PID_ES <sup>2)</sup>	
		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.

### Note

#### S7-1500 Motion Control

The technology objects TO\_OutputCam, TO\_CamTrack and TO\_MeasuringInput on S7-1500 are handled separately.

You can find further information in section "S7-1500 Motion Control (Page 339)".

### 7.18.13.3 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
	LIInt	long
	USInt	byte
	UInt	ushort
	UDint	uint
	ULLInt	ulong
	Floating-point numbers	Real
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

### 7.18.13.4 Querying the composition of technology objects

#### Requirement

- The Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- A PLC is determined in the project.  
See [Querying PLC and HMI targets \(Page 165\)](#)

#### 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)
{
    TechnologicalInstanceDBGGroup technologicalObjectGroup =
    plcSoftware.TechnologicalObjectGroup;
    TechnologicalInstanceDBComposition technologicalObjects =
    technologicalObjectGroup.TechnologicalObjects;
}
```

#### See also

[Standard libraries \(Page 40\)](#)

### 7.18.13.5 Creating technology object

#### Requirement

- The Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- A PLC is determined in the project.  
See [Querying PLC and HMI targets \(Page 165\)](#)

#### Application

Only technology objects that are listed in the section [Overview of technology objects and versions \(Page 320\)](#) can be created. An exception is thrown for unsupported technology objects or invalid parameters.

**Note****S7-1500 Motion Control**

The technology objects TO\_OutputCam, TO\_CamTrack and TO\_MeasuringInput on S7-1500 are handled separately.

You can find further information in section "S7-1500 Motion Control (Page 339)".

---

**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 320).

**See also**

Standard libraries (Page 40)

**7.18.13.6 Deleting technology object****Requirement**

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- A PLC is determined in the project.  
See Querying PLC and HMI targets (Page 165)
- The technology object exists.  
See Finding technology object (Page 327)

## 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 40)

### 7.18.13.7 Compiling technology object

#### Requirement

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- A PLC is determined in the project.  
See Querying PLC and HMI targets (Page 165)
- The technology object exists.  
See Creating technology object (Page 323)

#### 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)
{
    TechnologicalInstanceDBGGroup 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 116)".

### See also

Standard libraries (Page 40)

### 7.18.13.8 Enumerating technology object

#### Requirement

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- A PLC is determined in the project.  
See Querying PLC and HMI targets (Page 165)

## 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 40)

### 7.18.13.9 Finding technology object

#### Requirement

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- A PLC is determined in the project.  
See Querying PLC and HMI targets (Page 165)

## 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 40)

### 7.18.13.10 Enumerating parameters of technology object

#### Requirement

- The Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- A PLC is determined in the project.  
See [Querying PLC and HMI targets \(Page 165\)](#)
- A technology object exists.  
See [Creating technology object \(Page 323\)](#) or [Finding parameters of technology object \(Page 328\)](#)
- The technology object ([Page 320](#)) 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 \(Page 40\)](#)

[Finding technology object \(Page 327\)](#)

### 7.18.13.11 Finding parameters of technology object

#### Requirement

- The Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)



- A PLC is determined in the project.  
See Querying PLC and HMI targets (Page 165)
- A technology object exists.  
See Creating technology object (Page 323)
- The technology object (Page 320) 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 331)

Parameters of S7-1500 Motion Control (Page 339)

Parameters of PID Control (Page 357)

Parameters of Counting (Page 358)

Parameters of Easy Motion Control (Page 358)

## See also

Standard libraries (Page 40)

Finding technology object (Page 327)

### 7.18.13.12 Reading parameters of technology object

#### Requirement

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## 7.18 Functions for accessing the data of a PLC device

- A PLC is determined in the project.  
See Querying PLC and HMI targets (Page 165)
- A technology object exists.  
See Creating technology object (Page 323)
- The technology object (Page 320) 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 40)
- Finding technology object (Page 327)

### 7.18.13.13 Writing parameters of technology object

#### Requirement

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- A PLC is determined in the project.  
See Querying PLC and HMI targets (Page 165)
- A technology object exists.  
See Creating technology object (Page 323)
- The technology object (Page 320) 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 331)

Parameters of S7-1500 Motion Control (Page 339)

Parameters of PID Control (Page 357)

Parameters of Counting (Page 358)

Parameters of Easy Motion Control (Page 358)

## See also

Standard libraries (Page 40)

Finding technology object (Page 327)

### 7.18.13.14 S7-1200 Motion Control

## Changing version of openness engineering library

If you use "Openness\PublicAPI\V14 SP1\Siemens.Engineering.dll" with TIA Portal V15, your existing openness application will still work.

If you change to "Openness\PublicAPI\V15\Siemens.Engineering.dll" with TIA Portal V15, you have to adapt all accesses to array tags for S7-1200 Motion Control.

The affected arrays for TO\_PositioningAxis are listed in the following table:

Access in Openness < V15	Access in Openness ≥ V15
_Sensor.Sensor[1].<all tags>	_Sensor[1].<all tags>
ControlPanel.Input.Command.Command[1].<all tags>	ControlPanel.Input.Command[1].<all tags>
ControlPanel.Output.Command.Command[1].<all tags>	ControlPanel.Output.Command[1].<all tags>
Internal.Internal[n].<all tags>	Internal[n].<all tags>
Sensor.Sensor[1].<all tags>	Sensor[1].<all tags>
StatusSensor.StatusSensor[1].<all tags>	StatusSensor[1].<all tags>

The affected arrays for TO\_CommandTable are listed in the following table:

Access in Openness < V15	Access in Openness ≥ V15
Command.Command[n].<all tags>	Command[n].<all tags>

## Connecting PROFIdrives by hardware address

### Requirement

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74).
- A project is open.  
See Opening a project (Page 97).
- 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 323).

## 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 adress 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 74).
- A project is open.  
See Opening a project (Page 97).
- 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 323).

## 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 adress 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 74).
- A project is open.  
See Opening a project (Page 97).
- 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 323).

## 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 74).
- A project is open.  
See Opening a project (Page 97).
- 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 323).

## 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 74).
- A project is open.  
See Opening a project (Page 97).
- 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 323).

## 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 74).
- A project is open.  
See Opening a project (Page 97).
- 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 323).

### 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/109754206>).

---

#### Note

In TIA Portal in the Parameter view of the technology object configuration you can find the column "Name in Openness".

---

### 7.18.13.15 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 74).
- A project is open.  
See Opening a project (Page 97).
- 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));
}

```

**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. In Openness the directly mapped parameters have the same order as in the "data navigation" in the parameter view of the technology object. After the directly mapped parameters the additional parameters follow in order of the table.

## 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

You can find additional information about the directly mapped parameters in the appendix of:

- SIMATIC S7-1500 Motion Control function manual:  
<https://support.industry.siemens.com/cs/ww/en/view/109749262> (<https://support.industry.siemens.com/cs/ww/en/view/109749262>)
- SIMATIC S7-1500T Motion Control function manual:  
<https://support.industry.siemens.com/cs/ww/en/view/109749263> (<https://support.industry.siemens.com/cs/ww/en/view/109749263>)
- SIMATIC S7-1500T Kinematics Functions function manual:  
<https://support.industry.siemens.com/cs/ww/en/view/109749264> (<https://support.industry.siemens.com/cs/ww/en/view/109749264>)

Some technology parameters that map to read-only data block tags need to be made writeable in the PublicAPI. The allowed values are the same ones as for the underlying data block tags. The affected parameters are listed in the following tables:

Name in Openness	Data type	TO_SpeedAxis	TO_PositioningAxis	TO_SynchronousAxis	TO_ExternalEncoder
Actor.Type	int	X	X	X	-
Actor.Interface.EnableDriveOutput	bool	X	X	X	-
Actor.Interface.DriveReadyInput	bool	X	X	X	-
Actor.DataAdaptionOffline	bool	X	X	X	-
VirtualAxis.Mode	uint	X	X	X	-
Sensor[n].DataAdaptionOffline <sup>1)</sup>	bool	-	X	X	-
Sensor[n].Existent <sup>1)</sup>	bool	-	X	X	-
Sensor[n].Interface.Number <sup>1)</sup>	uint	-	X	X	-
Sensor[n].Type <sup>1)</sup>	int	-	X	X	-
Sensor.DataAdaptionOffline	bool	-	-	-	X
Sensor.Interface.Number	uint	-	-	-	X
Sensor.Type	int	-	-	-	X

Name in Openness	Data type	TO_OutputCam	TO_MeasuringInput	TO_Kinematics <sup>2)</sup>
Interface.LogicOperation	int	X	-	-
Parameter.MeasuringInputType	int	-	X	-

## 7.18 Functions for accessing the data of a PLC device

Name in Openness	Data type	TO_OutputCam	TO_MeasuringInput	TO_Kinematics <sup>2)</sup>
Kinematics.TypeOfKinematics	int	-	-	X
MotionQueue.MaxNumberOfCommands	int	-	-	X

1) S7-1500 PLC:  $n=1$ ; S7-1500T PLC:  $1 \leq n \leq 4$

2) S7-1500T PLC

## Parameters not mapped directly to technology object data block tags

For S7-1500 Motion Control technology objects the following additional parameters which do not directly map to data block tags are available:

Name in Openness	Name in function view	Possible value	Data type in Openness	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_External-Encoder
_Properties.MotionType	Axis type respectively "Technological unit of the position"	0: Linear 1: Rotary	int	-	X	X
_Units.LengthUnit	Position units	See tag Units.LengthUnit <sup>3)</sup>	uint	-	X	X
_Units.VelocityUnit	Velocity units	See tag Units.VelocityUnit <sup>3)</sup>	uint	X	X	X
_Units.TorqueUnit	Torque units	See tag Units.TorqueUnit <sup>3)</sup>	uint	X	X	-
_Units.ForceUnit	Force units	See tag Units.ForceUnit <sup>3)</sup>	uint	-	X	-
_Actor.Interface.Telegram	Drive telegram	Telegram number <sup>4)</sup>	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	-
_Sensor[n].Interface.Telegram <sup>5)</sup>	Encoder telegram	Telegram number <sup>4)</sup>	uint	-	X	-
_Sensor[n].Active-Homing.DigitalInputAddress <sup>5)</sup>	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_Sensor[n].Passive-Homing.DigitalInputAddress <sup>5)</sup>	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_External-Encoder
_PositionLimits_HW.MinSwitch-Address	Hardware low limit switch input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_PositionLimits_HW.MaxSwitch-Address	Hardware high limit switch input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_Sensor.Interface.Telegram	Encoder telegram	Telegram number <sup>4)</sup>	uint	-	-	X
_Sensor.Passive-Homing.DigitalInputAddress	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	-	X

For output cam, cam track and measuring input technology objects the following additional parameter is available:

Name in Openness	Name in function view	Possible value	Data type
_AssociatedObject	Associated object	PublicAPI-object	SW.TechnologicalObjects.TechnologicalInstanceDB

For kinematics technology object the following additional parameters are available (S7-1500T):

Name in Openness	Name in function view	Possible value	Data type
_KinematicsAxis[1...4]	Axis 1 - 3, Orientation axis	Axis that can be connected to TO_Kinematics objects	SW.TechnologicalObjects.TechnologicalInstanceDB
_Units.LengthUnit	Units of measurement > Position	See tag Units.LengthUnit <sup>3)</sup>	uint
_Units.LengthVelocityUnit	Units of measurement > Velocity	See tag Units.LengthVelocityUnit <sup>3)</sup>	uint
_Units.AngleUnit	Units of measurement > Angle	See tag Units.AngleUnit <sup>3)</sup>	uint
_Units.AngleVelocityUnit	Units of measurement > Angle velocity	See tag Units.AngleVelocityUnit <sup>3)</sup>	uint

3) possible values are described in the function manual S7-1500 Motion Control on chapter units tags (TO)

4) possible values are described in the function manual S7-1500 Motion Control on chapter PROFIdrive telegrams

5) S7-1500 PLC: n=1; S7-1500T PLC: 1≤n≤4

**Program code: Directly mapped data block tags**

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;
}
```

**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 manual:  
<https://support.industry.siemens.com/cs/ww/en/view/109749262> (<https://support.industry.siemens.com/cs/ww/en/view/109749262>)
- SIMATIC S7-1500T Motion Control function manual:  
<https://support.industry.siemens.com/cs/ww/en/view/109749263> (<https://support.industry.siemens.com/cs/ww/en/view/109749263>)
- SIMATIC S7-1500T Kinematics Functions function manual:  
<https://support.industry.siemens.com/cs/ww/en/view/109749264> (<https://support.industry.siemens.com/cs/ww/en/view/109749264>)



## Connecting drives

### Requirement

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74).
- A project is open.  
See Opening a project (Page 97).
- A S7-1500 PLC is determined in the project.
- 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. 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 one module.
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(HW.Channel channel)	Connects to a channel
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Connects specifying bit addresses directly
void Connect(string pathToDBMember)	Connects to a data block tag
void Connect(SW.Tags.PlcTag outputTag)	Connects to a PLC tag
void Disconnect()	Disconnects an existing connection

#### Note

##### Automatic connections

Note that the same behavior as in the user interface also applies here. Whenever the actor interface is connected via one of the connection methods and the telegram contains a sensor part or telegram 750. These parts are connected automatically.

## 7.18 Functions for accessing the data of a PLC device

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: <ul style="list-style-type: none"> <li>• Default Only modules that are recognized as valid connection partners can be selected.</li> <li>• AllowAllModules Corresponds to selecting "Show all modules" in the user interface.</li> </ul>
Channel	HW.Channel	Connected channel
PathToDBMember	string	Connected technology object data block tag
OutputTag	SW.Tags.PlcTag	Connected PLC tag (analog connection)
SensorIndexInActor-Telegram	int	Connected sensor part 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.

**Note****Access the sensor interface**

To access the sensor interface you can use `SensorInterface[m]` with  $0 \leq m \leq 3$ .

**Program Code: void Connect(HW.DeviceItem moduleInOut)**

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 UseServiceAxisHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, DeviceItem devItem)
{
    //Retrieve service AxisHardwareConnectionProvider
    AxisHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<AxisHardwareConnectionProvider>();

    //Connect ActorInterface with DeviceItem
    connectionProvider.ActorInterface.Connect(devItem);

    //Connect first SensorInterface with DeviceItem
    connectionProvider.SensorInterface[0].Connect(devItem);

    //Check ConnectionState of ActorInterface
    bool actorInterfaceConnectionState = connectionProvider.ActorInterface.IsConnected;

    //Check ConnectionState of first SensorInterface
    bool sensorInterfaceConnectionState =
    connectionProvider.SensorInterface[0].IsConnected;
}
```

**Connecting telegram 750****Requirement**

- The Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- A 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.Deviceltem moduleInOut)	Connects to input and output addresses at one module
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut)	Connects to input and output addresses at separate modules
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem 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
void Connect(string pathToDBMember)	Connects to a data block tag
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
InputOutputModule	HW.Deviceltem	Connected module that contains input and output addresses
InputModule	HW.Deviceltem	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.Deviceltem	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

Attribute	Data type	Description
ConnectOption	ConnectOption	Value of the ConnectOption that has been set when the connection was made: <ul style="list-style-type: none"> <li>• Default Only modules that are recognized as valid connection partners can be selected.</li> <li>• AllowAllModules Corresponds to selecting "Show all modules" in the user interface.</li> </ul>
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 74).
- A project is open.  
See Opening a project (Page 97).
- A 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
<code>void Connect(HW.Deviceltem moduleInOut)</code>	Connects to input and output addresses at one module.
<code>void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut)</code>	Connects to input and output addresses at separate modules.
<code>void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut, ConnectOption connectOption)</code>	Connects to input and output addresses at separate modules, specifying an additional <code>ConnectOption</code>
<code>void Connect(HW.Channel channel)</code>	Connects to a channel
<code>void Connect(int addressIn, int addressOut, ConnectOption connectOption)</code>	Connects specifying bit addresses directly
<code>void Connect(string pathToDBMember)</code>	Connects to a data block tag
<code>void Connect(SW.Tags.PlcTag outputTag)</code>	Not relevant for connecting encoders
<code>void Disconnect()</code>	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
<code>IsConnected</code>	bool	TRUE: Interface is connected FALSE: Interface is not connected
<code>InputOutputModule</code>	HW.Deviceltem	Connected module that contains input and output addresses
<code>InputModule</code>	HW.Deviceltem	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.
<code>OutputModule</code>	HW.Deviceltem	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.
<code>InputAddress</code>	int	Logical input address of connected object, for example 256.
<code>OutputAddress</code>	int	Logical output address of connected object, for example 256.
<code>ConnectOption</code>	ConnectOption	Value of the <code>ConnectOption</code> that has been set when the connection was made: <ul style="list-style-type: none"> <li>• <code>Default</code> Only modules that are recognized as valid connection partners can be selected.</li> <li>• <code>AllowAllModules</code> Corresponds to selecting "Show all modules" in the user interface.</li> </ul>
<code>Channel</code>	HW.Channel	Connected channel
<code>PathToDBMember</code>	string	Connected data block tag

Attribute	Data type	Description
OutputTag	SW.Tags.PlcTag	Not relevant for connecting encoders
SensorIndexInActor-Telegram	int	<p>Connected sensor telegram</p> <p>The attribute is only relevant for sensor interfaces.</p> <p>0: Encoder is not connected</p> <p>1: Encoder is connected to first sensor interface in telegram</p> <p>2: Encoder is connected to second sensor interface in telegram</p> <p>For the actor interface the value is always 0.</p>

### 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 74).
- A project is open.  
See Opening a project (Page 97).
- 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
<code>void Connect(HW.Channel channel)</code>	Connects to a channel
<code>void Connect(SW.Tags.PlcTag outputTag)</code>	Connects to a PLC tag
<code>void Connect(int address)</code>	Connects specifying bit addresses directly
<code>void Disconnect()</code>	Disconnects an existing connection

You can use the following read-only attributes to determine how the technology object is connected:

Attribute	Data type	Description
<code>IsConnected</code>	<code>bool</code>	TRUE: Technology object is connected FALSE: Technology object is not connected
<code>Channel</code>	<code>HW.Channel</code>	Connected channel
<code>OutputTag</code>	<code>SW.Tags.PlcTag</code>	Connected PLC tag
<code>OutputAddress</code>	<code>int</code>	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;
}
```



## Connecting measuring inputs to hardware

### Requirement

- The Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74).
- A project is open.  
See Opening a project (Page 97).
- 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
<code>void Connect(HW.Channel channel)</code>	Connects to a channel
<code>void Connect(HW.DeviceItem moduleIn, int channelIndex)</code>	Connects to a module, specifying an additional channel index
<code>void Connect(int address)</code>	Connects specifying bit addresses directly
<code>void Disconnect()</code>	Disconnects an existing connection

You can use the following read-only attributes to determine how the technology object is connected:

Attribute	Data type	Description
<code>IsConnected</code>	<code>bool</code>	TRUE: Technology object is connected FALSE: Technology object is not connected
<code>InputModule</code>	<code>HW.DeviceItem</code>	Connected module that contains input addresses
<code>ChannelIndex</code>	<code>int</code>	Index of connected channel with respect to <code>InputModule</code>
<code>Channel</code>	<code>HW.Channel</code>	Connected channel
<code>InputAddress</code>	<code>int</code>	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 74).
- A project is open. See Opening a project (Page 97).
- A S7-1500 PLC is determined in the project.
- 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\_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 succesful FALSE: Disconnection was not succesful

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

### 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);
}
```

## 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 74).
- A project is open.  
See Opening a project (Page 97).
- A S7-1500 PLC is determined in the project.  
See Querying PLC and HMI targets (Page 165)
- The technology object exists.

### Application

To export or import the data of a technology object cam 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.

Method	Description
<code>void SaveCamDataBinary(System.IO.FileInfo destinationFile)</code>	Exports the data in binary format in the destination file.
<code>void SaveCamDataPointList(System.IO.FileInfo destinationFile, CamDataFormatSeparator separator, int samplePoints)</code>	Exports the data in format "PointList" in the destination file.
<code>void SaveCamData(System.IO.FileInfo destinationFile, CamDataFormat format, CamDataFormatSeparator separator)</code>	Exports the 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.
<code>void LoadCamData(System.IO.FileInfo sourceFile, CamDataFormatSeparator separator)</code>	Imports the cam data in the format "MCD", "SCOUT" or "Pointlist" to the project.
<code>void LoadCamDataBinary(System.IO.FileInfo sourceFile)</code>	Imports the cam data from a binary file to the project.

You can use the following attributes:

Attribute	Data type	Description
separator	CamDataFormatSeparator	Allowed values <ul style="list-style-type: none"> <li>• tab</li> <li>• comma</li> </ul>
samplePoints	int	Number of interpolation points that should be exported.
format	CamDataFormat	Allowed values <ul style="list-style-type: none"> <li>• MCD</li> <li>• SCOUT</li> <li>• Pointlist</li> </ul>

Attribute	Data type	Description
destinationFile	System.IO.FileInfo	Name of destination 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: Export cam data

Modify the following program code to export 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: Import cam data

Modify the following program code to import 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);
}
```

#### 7.18.13.16 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 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

### 7.18 Functions for accessing the data of a PLC device

- 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 in SIMATIC S7-1200/S7-1500 PID control function manual on the internet (<https://support.industry.siemens.com/cs/ww/en/view/108210036>).

#### 7.18.13.17 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>).

#### 7.18.13.18 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).

## 7.18.14 Tags and Tag tables

### 7.18.14.1 Starting the "PLC Tags" editor

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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 417)

### 7.18.14.2 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 74)
- A project is open.  
See Opening a project (Page 97)
- A PlcSoftware instance was retrieved from a PLC device item.  
See Querying PLC and HMI targets (Page 165)

#### 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;
}
```

### 7.18.14.3 Creating PLC tag table

#### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- A PlcSoftware instance was retrieved from a PLC device item.  
See Querying PLC and HMI targets (Page 165)

#### 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 165)

**7.18.14.4 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 74)
- A project is open.  
See Opening a project (Page 97)
- A PlcSoftware instance was retrieved from a PLC device item.  
See Querying PLC and HMI targets (Page 165)

**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
}
```

**7.18.14.5 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

**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");
}
```

### 7.18.14.6 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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();
    }
}
```

### 7.18.14.7 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 74)
- A project is open.  
See Opening a project (Page 97)

**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
}
```

**7.18.14.8 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

**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");
}
```

### 7.18.14.9 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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;
}
```

### 7.18.14.10 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 74)
- A project is open.  
See Opening a project (Page 97)

## 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();
    }
}
```

### 7.18.14.11 Enumerating PLC tags

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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
    }
}
```

### 7.18.14.12 Accessing PLC tags

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## 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

```
//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();
    }
}
```

#### 7.18.14.13 Accessing PLC constants

##### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

##### 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 362)

Deleting user-defined groups for PLC tags (Page 363)

Deleting a PLC tag table from a group (Page 366)

Accessing PLC tags (Page 367)

Starting the "PLC Tags" editor (Page 359)

Reading the time of the last changes of a PLC tag table (Page 366)

## 7.19 Functions on OPC

### 7.19.1 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 74)
- A project is open  
See Opening a project (Page 97)

#### 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.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <OPCUA_Server_WinCC xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ua="http://opcfounde
3
4  <SecuredApplication>
5    <BaseAddresses>
6      <ua:String>opc.tcp://[HostName]:4861</ua:String>
7    </BaseAddresses>
8    <SecurityProfileUris>
9      <SecurityProfile>
10       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#None</ProfileUri>
11       <Enabled>true</Enabled>
12     </SecurityProfile>
13     <SecurityProfile>
14       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
15       <Enabled>true</Enabled>
16     </SecurityProfile>
17     <SecurityProfile>
18       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
19       <Enabled>true</Enabled>
20     </SecurityProfile>
21     <SecurityProfile>
22       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
23       <Enabled>true</Enabled>
24     </SecurityProfile>
25   </SecurityProfileUris>
26 </SecuredApplication>
27
28 <ServerConfiguration>
29   <SecurityPolicies>
30     <SecurityPolicy>
31       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#None</ProfileUri>
32       <MessageSecurityModes>None</MessageSecurityModes>
33     </SecurityPolicy>
34     <SecurityPolicy>
35       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
36       <MessageSecurityModes>Sign</MessageSecurityModes>
37     </SecurityPolicy>
38     <SecurityPolicy>
39       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
40       <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
41     </SecurityPolicy>
42     <SecurityPolicy>
43       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
44       <MessageSecurityModes>Sign</MessageSecurityModes>
45     </SecurityPolicy>
46     <SecurityPolicy>
47       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
48       <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
49     </SecurityPolicy>
50     <SecurityPolicy>
51       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
52       <MessageSecurityModes>Sign</MessageSecurityModes>
53     </SecurityPolicy>
54     <SecurityPolicy>
55       <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
56       <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
57     </SecurityPolicy>
58   </SecurityPolicies>

```

**See also**

Connecting to the TIA Portal (Page 74)

Opening a project (Page 97)

## 7.19.2 Setting OPC UA security policy

### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
Opening a project (Page 97)
- OPC UA server is activated

### Application

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, you will encounter `EngineeringNotSupportedException` 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	OpcUaSecurityPolicies-None	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	

## Program code

Modify the following program code to set the security policy in OPC UA using TIA Portal Openness:

```
DeviceItem UpcUaSubmodule= ...; "  
object SecurityPolicies = UpcUaSubmodule.GetAttribute("OpcUaSecurityPolicies");  
if(SecurityPolicies | OpcUaSecurityPolicies.OpcUaSecurityPolicies256S ==  
OpcUaSecurityPolicies.OpcUaSecurityPolicies256S)  
{  
  //Do something  
}  
UpcUaSubmodule.SetAttribute("OpcUaSecurityPolicies",  
OpcUaSecurityPolicies.OpcUaSecurityPolicies256S |  
OpcUaSecurityPolicies.OpcUaSecurityPolicies256SHASE);
```

## See also

[Connecting to the TIA Portal \(Page 74\)](#)

[Opening a project \(Page 97\)](#)

## **7.20 SiVArc Openness**

### **7.20.1 Introduction**



## 7.21 Openness for CP 1604/CP 1616/CP 1626

### General

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.

### Requirements

- The TIA Portal Openness application is connected to the TIA Portal. See "Establishing a connection to the TIA Portal".
- A project is open. See "Open project".
- To compile the project, all devices must be "offline".

### Configuration of transfer areas

#### Creating transfer areas

For example, to create a "CD" type transfer area for a CP 1604, use the following program code:

```
NetworkInterface cpItf = CP
1604Interface.GetService<NetworkInterface>();

// Create TransferAreas

TransferAreaComposition transferAreas = cpItf.TransferAreas;

// Simple TransferArea of type Input

TransferArea transferAreaInput =
    transferAreas.Create("Input CD", TransferAreaType.CD);
```

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 PROFI-safe
	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, use the following program code, for example:

```
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()". Use the following program code, for example:

```
const string myIndividualComment = "MyIndividualComment";
transferAreaTm.SetAttribute("Comment", myIndividualComment);
Int32 updateTime = transferAreaTm.GetAttribute("TransferUpdateTime")
```

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 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, use the following program code:

```
transferAreaInput.Delete();
```

### Iteration via transfer areas

To iterate transfer areas, use the following program code:

```

TransferAreaComposition transferAreas = cpItf.TransferAreas;
    foreach (TransferArea transferArea in transferAreas)
    {
        transferArea.Delete();
    }

```

## Configuration of IO routing

### Creating IO routes

To create IO routes, use 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, use the following program code:

```
transferAreaMappingRule.Delete();
```

### Iteration via IO routing

To iterate via IO routing, use the following program code:

```
TransferAreaMappingRuleComposition routingTable =  
    transferArea.TransferAreaMappingRules;  
foreach (TransferAreaMappingRule route in routingTable)  
{  
    route.Delete();  
}
```

## 7.22 Openness for SIMATIC Ident

### 7.22.1 Openness for SIMATIC Ident

SIMATIC Ident devices (communication modules, RF600 reader and MV400/MV500 readers) can be configured with TIA Portal Openness. The following sections provide an overview of the module-specific parameters.

#### Basic knowledge required

The following passages assume you have general knowledge of automation engineering and identification systems, as well as TIA Portal Openness.

#### Additional information

You can find detailed information on TIA Portal Openness in the help "Automating projects with scripts". This help contains concrete program examples (see sections "Functions of projects and project data" and "Functions for accessing devices, networks and connections").

#### Requirement

- The Openness application is connected to the TIA Portal.
- A project is open.
- All devices must be offline to compile the project.

### 7.22.2 ASM 456

The Openness parameters for the "ASM 456" module are described below.

Table 7-1 Parameters of the "ASM 456" module; "Word: 2 IN/OUT DP-V1" module

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
User mode	UserMode	ulong	Ident profile/RFID standard profile	3	R/W	Selection depends on the communications module and Ident system being used. With this parameter, you select the block: <ul style="list-style-type: none"> <li>• Ident profile/RFID standard profile: The program block for the Ident profile/RFID standard profile is used on the controller.</li> <li>• FB 45 / FC 45: Single tag mode FB 45 (PROFIBUS/PROFINET) or FC 45 (PROFIBUS) is used on the controller.</li> <li>• FB 55 / FC 55: Multitag mode. FB 55 (PROFIBUS/PROFINET) or FC 55 (PROFIBUS) is used on the controller.</li> <li>• FC 56 File handler for S7-300 and S7-400</li> </ul>
			FB 45 / FC 45	1	R/W	
			FB 55 / FC 55	4	R/W	
			FC 56	5	R/W	

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
MOBY mode	MobyMode	ulong	RF200/ RF300/ RF600; MV4x0; MO- BY U/D	5	R/W	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set the mode of the communications module.</p> <ul style="list-style-type: none"> <li>RF200/RF300/RF600; MV4x0; MOBY U/D</li> <li>MOBY I/E normal addressing</li> <li>RF300 filehandler</li> <li>MOBY U file handler</li> <li>MOBY I file handler</li> </ul> <p>Normal addressing: The transponder is addressed with physical addresses. Filehandler: Prior to use, the transponder needs to be formatted.</p>
			MOBY I/E normal addressing	1	R/W	
			RF300 filehandler	149	R/W	
			MOBY U file handler	133	R/W	
			MOBY I file handler	129	R/W	
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set the transmission speed between the communications module and reader.</p> <p>When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).</p> <p>When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.</p>
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
Diagnostics messages	Diagnostic-Messages	int	None	1	R/W	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set whether hardware diagnostics messages will be reported.</p> <ul style="list-style-type: none"> <li>• None: Apart from standard diagnostics, no other alarms are generated.</li> <li>• Hard errors: Critical hardware errors are reported by the S7 diagnostics.</li> <li>• Hard/soft errors low priority Critical hardware errors and errors that occur during command processing are reported by the S7 diagnostics. The "Ext_Diag" bit is not set.</li> <li>• Hard/soft errors high priority Critical hardware errors and errors that occur during command processing are reported by the S7 diagnostics. The "Ext_Diag" bit is set.</li> </ul>
			Hard errors	2	R/W	
			Hard/soft errors low priority	3	R/W	
			Hard/soft errors high priority	5	R/W	
Suppression of Error LED	SuppressionOfErrorLed	int	None	0	R/W	<p>Disabling the Error LED (ERR) of a channel.</p> <p>The communications module has two channels to which the readers / optical readers can be connected. The Error LED of the other channel flashes permanently when only one of the channels is being used. With the help of the suppression, you can disable the Error LED of the unused channel.</p>
			Channel 1	1	R/W	
			Channel 2	2	R/W	



Table 7-2 Parameters of the "ASM 456" module; "RF680R/RF685R" module

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
User mode	UserMode	ulong	Ident profile/ RFID standard profile	3	R	Selection depends on the communications module and Ident system being used. With this parameter, you select the block: <ul style="list-style-type: none"> <li>Ident profile/RFID standard profile: The program block for the Ident profile/RFID standard profile is used on the controller.</li> </ul>
MOBY mode	MobyMode	ulong	RF680R/ RF685R	32	R	Selection depends on the communications module and Ident system being used. With this parameter, you set the mode of the communications module. <ul style="list-style-type: none"> <li>RF680R/RF685R Normal addressing: The transponder is addressed with physical addresses. Filehandler: Prior to use, the transponder needs to be formatted.</li> </ul>
Transmission speed	BaudRate	ulong	115.2 kBd	14	R	Selection depends on the communications module and Ident system being used. With this parameter, you set the transmission speed between the communications module and reader. When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power). When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
Diagnostics messages	Diagnostic-Messages	int	None	1	R/W	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set whether hardware diagnostics messages will be reported.</p> <ul style="list-style-type: none"> <li>• None: Apart from standard diagnostics, no other alarms are generated.</li> <li>• Hard errors: Critical hardware errors are reported by the S7 diagnostics.</li> <li>• Hard/soft errors low priority Critical hardware errors and errors that occur during command processing are reported by the S7 diagnostics. The "Ext_Diag" bit is not set.</li> <li>• Hard/soft errors high priority Critical hardware errors and errors that occur during command processing are reported by the S7 diagnostics. The "Ext_Diag" bit is set.</li> </ul>
			Hard errors	2	R/W	
			Hard/soft errors low priority	3	R/W	
			Hard/soft errors high priority	5	R/W	
Suppression of Error LED	SuppressionOfErrorLed	int	None	0	R/W	<p>Disabling the Error LED (ERR) of a channel.</p> <p>The communications module has two channels to which the readers / optical readers can be connected. The Error LED of the other channel flashes permanently when only one of the channels is being used. With the help of the suppression, you can disable the Error LED of the unused channel.</p>
			Channel 1	1	R/W	
			Channel 2	2	R/W	

### 7.22.3 ASM 475

The Openness parameters for the "ASM 475" module are described below.

Table 7-3 Parameters of the module "ASM 475"

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
MOBY mode	MobyMode	ulong	MOBY I/E/F normal addressing	1	R/W	<p>Selection depends on the communications module and Ident system being used.</p> <p>With this parameter, you set the mode of the communications module.</p> <ul style="list-style-type: none"> <li>MOBY I/E/F normal addressing</li> <li>MOBY I file handler</li> <li>RF200/RF300/RF600; MOBY U/D normal addressing</li> <li>MOBY U file handler</li> </ul> <p>Normal addressing: The transponder is addressed with physical addresses.</p> <p>Filehandler: Prior to use, the transponder needs to be formatted.</p>
			MOBY I file handler	129	R/W	
			RF200/RF300/RF600; MOBY U/D normal addressing	5	R/W	
			MOBY U file handler	133	R/W	
Transmission speed	BaudRate	ulong	19200 Bd	9	R/W	<p>Selection depends on the communications module and Ident system being used.</p> <p>With this parameter, you set the transmission speed between the communications module and reader.</p> <p>When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).</p> <p>When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.</p>
			57600 Bd	13	R/W	
			115200 Bd	14	R/W	

## 7.22.4 RF120C

The Openness parameters for the "RF120C" module are described below.

Table 7-4 Parameters of the module "RF120C"; parameters of the parameter group "Readers"

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
Diagnostics messages	Diagnostic-Messages	int	None	1	R/W	With this parameter, you set whether hardware diagnostics messages will be reported. <ul style="list-style-type: none"> <li>• None: Apart from standard diagnostics, no other alarms are generated.</li> <li>• Hard errors: Critical hardware errors are reported by the S7 diagnostics.</li> </ul>
			Hard errors	2	R/W	
User mode	UserMode	ulong	Ident profile	3	R	Selection depends on the communications module and Ident system being used. With this parameter, you select the block: <ul style="list-style-type: none"> <li>• Ident profile The program block for the Ident profile is used on the controller.</li> </ul>
Ident device / system	IdentDeviceOrSystem	int	RF200 general	53	R/W	Selection of the connected Ident device / system. Depending on the selection you make, the "Ident system" parameter group is adapted.
			RF290R	69	R/W	
			RF300 general	85	R/W	
			RF380R	101	R/W	
			MOBY U	5	R/W	
			General Reader	197	R/W	
			Parameters via FB / optical readers	214	R/W	
			RF600 <sup>1)</sup>	117	R/W	
			SLG D10S <sup>1)</sup>	21	R/W	
			SLG D11S/ D12S <sup>1)</sup>	37	R/W	

<sup>1)</sup> The sub-parameters of these parameter are not supported by Openness.

The Openness parameters of the parameter group "Ident device/system" are described below.

Table 7-5 Parameters of the module "RF120C"; parameters of the parameter group "Ident device/system: RF200 general"

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	<p>Selection depends on the communications module and Ident system being used. Note that the value specified here is adopted automatically from the device configuration of the connected devices.</p> <p>With this parameter, you set the transmission speed between the communications module and reader.</p> <p>When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).</p> <p>When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.</p>
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	
Presence check	Presence-Check	ulong	On	1	R/W	<p>On = As soon as there is a transponder in the antenna field of the reader, its presence is reported.</p> <p>Off = The presence check on the FB is suppressed. The antenna on the reader is nevertheless turned on as long as it has not been turned off by a command.</p>
			Off	0	R/W	
Reset ERR LED	ResetErrorLED	int	On	2	R/W	<p>On = the flashing of the error LED on the communications module is reset by each FB reset.</p> <p>Off = the error LED always indicates the last error. The display can only be reset by turning off the communications module.</p>
			Off	0	R/W	

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Max. number of transponders	MaxNoOf-Transponders	int	1	<number>	R	Number of transponders expected in the antenna field. The selection depends on the connected device.
Transponder type	TransponderType	int	ISO 15693	1	R	Selection of the transponder types used. The selection depends on the connected device.

Table 7-6 Parameters of the module "RF120C"; parameters of the parameter group "Ident device/system: RF290R"

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	Selection depends on the communications module and Ident system being used. Note that the value specified here is adopted automatically from the device configuration of the connected devices. With this parameter, you set the transmission speed between the communications module and reader. When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power). When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	
Presence check	Presence-Check	ulong	On	1	R/W	On = As soon as there is a transponder in the antenna field of the reader, its presence is reported. Off = The presence check on the FB is suppressed. The antenna on the reader is nevertheless turned on as long as it has not been turned off by a command.
			Off	0	R/W	

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Reset ERR LED	ResetErrorLED	int	On	2	R/W	On = the flashing of the error LED on the communications module is reset by each FB reset.  Off = the error LED always indicates the last error. The display can only be reset by turning off the communications module.
			Off	0	R/W	
HF power	RfPower	double	0.50 ... 5.00	<text>	R/W	Setting for the output power of the reader.  The selectable values depend on the connected device.
Max. number of transponders	MaxNoOfTransponders	int	1	<number>	R	Number of transponders expected in the antenna field.  The selection depends on the connected device.
Transponder type	TransponderType	int	ISO 15693	1	R	Selection of the transponder types used. The selection depends on the connected device.

Table 7-7 Parameters of the module "RF120C"; parameters of the parameter group "Ident device/system: RF300 general"

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	Selection depends on the communications module and Ident system being used.  Note that the value specified here is adopted automatically from the device configuration of the connected devices.  With this parameter, you set the transmission speed between the communications module and reader.  When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).  When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Presence check	Presence-Check	ulong	On	1	R/W	<p>On = As soon as there is a transponder in the antenna field of the reader, its presence is reported.</p> <p>Off (RF field on) = the presence check in the FB is suppressed. The antenna on the reader is nevertheless turned on as long as it has not been turned off by a command.</p> <p>Off (RF field off) = the antenna is turned on only when a command is sent and it then turns itself off again.</p>
			Off (RF field on)	3	R/W	
			Off (RF field off)	2	R/W	
Reset ERR LED	ResetErrorLED	int	On	2	R/W	<p>On = the flashing of the error LED on the communications module is reset by each FB reset.</p> <p>Off = the error LED always indicates the last error. The display can only be reset by turning off the communications module.</p>
			Off	0	R/W	
Max. number of transponders	MaxNoOfTransponders	int	1	<number>	R	<p>Number of transponders expected in the antenna field.</p> <p>The selection depends on the connected device.</p>
Transponder type	TransponderType	int	RF300	5	R/W	<p>Selection of the transponder types used. The selection depends on the connected device.</p>
			ISO 15693	1	R/W	



Table 7-8 Parameters of the module "RF120C"; parameters of the parameter group "Ident device/system: RF380R"

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	<p>Selection depends on the communications module and Ident system being used. Note that the value specified here is adopted automatically from the device configuration of the connected devices.</p> <p>With this parameter, you set the transmission speed between the communications module and reader.</p> <p>When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).</p> <p>When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.</p>
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	
Presence check	Presence-Check	ulong	On	1	R/W	<p>On = As soon as there is a transponder in the antenna field of the reader, its presence is reported.</p> <p>Off (RF field on) = the presence check in the FB is suppressed. The antenna on the reader is nevertheless turned on as long as it has not been turned off by a command.</p> <p>Off (RF field off) = the antenna is turned on only when a command is sent and it then turns itself off again.</p>
			Off (RF field on)	3	R/W	
			Off (RF field off)	2	R/W	
Reset ERR LED	ResetErrorLED	int	On	2	R/W	<p>On = the flashing of the error LED on the communications module is reset by each FB reset.</p> <p>Off = the error LED always indicates the last error. The display can only be reset by turning off the communications module.</p>
			Off	0	R/W	
HF power	RfPower	double	0.50 ... 5.00	<text>	R/W	<p>Setting for the output power of the reader.</p> <p>The selectable values depend on the connected device.</p>

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Max. number of transponders	MaxNoOf-Transponders	int	1	<number>	R	Number of transponders expected in the antenna field. The selection depends on the connected device.
Transponder type	TransponderType	int	RF300	5	R/W	Selection of the transponder types used. The selection depends on the connected device.
			ISO 15693	1	R/W	

Table 7-9 Parameters of the module "RF120C"; parameters of the parameter group "Ident device/system: General Reader"

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	Selection depends on the communications module and Ident system being used. Note that the value specified here is adopted automatically from the device configuration of the connected devices.  With this parameter, you set the transmission speed between the communications module and reader.  When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).  When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	
Byte sequence of the reset parameter	ByteSequenceHex	string	<sup>1)</sup>	<text>	R/W	Byte sequence of the reset parameter of the reader.

<sup>1)</sup> For a detailed description of this parameter, see the following paragraph.

Table 7-10 Parameters of the module "RF120C"; parameters of the parameter group "Ident device/system: Parameters via FB / optical readers"

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	Selection depends on the communications module and Ident system being used. Note that the value specified here is adopted automatically from the device configuration of the connected devices. With this parameter, you set the transmission speed between the communications module and reader. When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power). When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	
MOBY mode	MobyMode	ulong	RF200/ RF300/ RF600; MV4x0; MO- BY U/D	5	R	With this parameter, you set the mode of the communications module.

### The parameter "Byte sequence of the reset parameter"

With this function, you can specify the Reset parameter using a byte array (16). This setting is intended only for trained users.

The following Reset parameters are available:

Table 7-11 Reset parameters

Byte	1	2...5	6	7...8	9	10	11	12	13...14	15	16
Value	4	0	10	0	scan- ning_ time	param	op- tion_1	distance_ limiting	multitag	field_on_ control	field_on_ time

### 7.22.5 RF170C

The Openness parameters for the "RF170C" module are described below.

Table 7-12 Parameters of the module "RF170C"; parameters of the parameter group "Module parameters"

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
User mode	UserMode	ulong	Ident profile/ RFID standard profile	3	R/W	Selection depends on the communications module and Ident system being used. With this parameter, you select the block: <ul style="list-style-type: none"> <li>Ident profile/RFID standard profile: The program block for the Ident profile/RFID standard profile is used on the controller.</li> <li>FB 45 / FC 45: Single tag mode FB 45 (PROFIBUS/PROFINET) or FC 45 (PROFIBUS) is used on the controller.</li> <li>FB 55 / FC 55: Multitag mode. FB 55 (PROFIBUS/PROFINET) or FC 55 (PROFIBUS) is used on the controller.</li> </ul>
			FB 45 / FC 45	1	R/W	
			FB 55 / FC 55	4	R/W	
MOBY mode	MobyMode	ulong	RF200/ RF300/ RF600; MV4x0; MO- BY U/D	5	R/W	Selection depends on the communications module and Ident system being used. With this parameter, you set the mode of the communications module. <ul style="list-style-type: none"> <li>RF200/RF300/RF600; MV4x0; MOBY U/D</li> <li>MOBY I/E</li> <li>MV3xx</li> <li>Freeport protocol</li> <li>RF300 filehandler</li> </ul> Normal addressing: The transponder is addressed with physical addresses. Filehandler: Prior to use, the transponder needs to be formatted.
			MOBY I/E	1	R/W	
			MV3xx	16	R/W	
			Freeport protocol	17	R/W	
RF300 filehandler	149	R/W				

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set the transmission speed between the communications module and reader.</p> <p>When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).</p> <p>When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.</p>
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	
Diagnostics messages	Diagnostic-Messages	int	None	1	R/W	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set whether hardware diagnostics messages will be reported.</p> <ul style="list-style-type: none"> <li>• None: Apart from standard diagnostics, no other alarms are generated.</li> <li>• Hard errors: Critical hardware errors are reported by the S7 diagnostics.</li> </ul>
			Hard errors	2	R/W	
Suppression of Error LED	SuppressionOfErrorLed	int	None	0	R/W	<p>Disabling the Error LED (ERR) of a channel.</p> <p>The communications module has two channels to which the readers / optical readers can be connected. The Error LED of the other channel flashes permanently when only one of the channels is being used. With the help of the suppression, you can disable the Error LED of the unused channel.</p>
			Channel 1	1	R/W	
			Channel 2	2	R/W	
Interface	ModuleInterface	int	RS232	1	R	Selection of the interface type that the connected hardware (reader / optical reader) uses.
			RS422	0	R	

Table 7-13 Parameters of the module "RF170C"; parameters of the parameter group "Freeport protocol"

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Data bits	DataBits	ulong	7	7	R/W	Selection of the number of bits on which a character is represented.
			8	8	R/W	
Parity	Parity	ulong	None	0	R/W	Parity selection A sequence of data bits can be expanded by a parity bit. With its value "0" or "1", the parity bit is added to the sum of all bits (data bits and parity bits) to form a defined status. This increases data reliability. <ul style="list-style-type: none"> <li>• None: Data is sent without a parity bit.</li> <li>• Odd: The parity bit is set so that the sum of the data bits (including the parity bit) is odd when the signal state is "1".</li> <li>• Even: The parity bit is set so that the sum of the data bits (including the parity bit) is even when the signal state is "1".</li> <li>• Fixed value 1: The parity bit is set permanently to the value "1".</li> <li>• Fixed value 0: The parity bit is set permanently to the value "0".</li> </ul>
			Odd	1	R/W	
			Even	2	R/W	
			Fixed value 1	3	R/W	
			Fixed value 0	7	R/W	
Stop bits	StopBits	ulong	1	1	R/W	Selection of the number of stop bits that indicate the end of a character. The stop bits are appended to every transferred character during transmission.
			2	2	R/W	

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
Specifying end detection	EndDetectionOfAReceivedFrame	ulong	After character delay time elapses	16	R/W	<p>Specifies the end detection of a received frame.</p> <ul style="list-style-type: none"> <li>After character delay time elapses: The frame has neither a fixed length nor defined end delimiters. The end of a frame is indicated by a gap in the character sequence. The size of this gap is specified by the character delay time.</li> <li>On receipt of fixed number of characters: The length of the received frame is always the same. When data is received, the end of the frame is recognized when the set number of characters has been received.</li> <li>On receipt of the end delimiter(s): At the end of the frame there are one or two defined end delimiters. When data is received, the end of the frame is recognized when the configured end delimiter(s) is/are received.</li> </ul>
			On receipt of fixed number of characters	32	R/W	
			On receipt of the end delimiter(s):	49	R/W	
No. of end delimiters	NumberOfEndDelimiters	ulong	1	1	R/W	<p>Selection of the number of end delimiters.</p> <p>A maximum of 2 end delimiters can be configured. When data is received, the end of the frame is recognized when the selected end delimiter combination is received.</p>
			2	2	R/W	

UI parameter	Openness parameter	Data type in Openness	UI parameter value	Openness parameter value	Default access	Description
1st end delimiter	FirstEndDelimiterReceiver	ulong	0...7F / 0...FF	<text>	R/W	Entry of the 1st end delimiter of maximum two end delimiters for end criteria "On receipt of the end delimiter(s)". The selected end delimiter or the selected end delimiter combination limits the length of the frame.  Parameter value depending on the "Data bits" parameter.
2nd end delimiter	SecondEndDelimiterReceiver	ulong	0...7F / 0...FF	<text>	R	Entry of the 2nd end delimiter of maximum two end delimiters for end criteria "On receipt of the end delimiter(s)". The selected end delimiter or the selected end delimiter combination limits the length of the frame.  Parameter value depending on the "Data bits" parameter.
Frame length	FrameLength	ulong	1...233 / 1...229	<text>	R	Entry of the frame length in bytes for the end criterion "On receipt of fixed number of characters".
Character delay time	CharacterDelayTime	ulong	0...65535	<text>	R	Entry of the time that may elapse until a frame end is recognized. Select the character delay time dependent on the send behavior of your communications partner. Depending on the data transmission speed the character delay time is limited to a minimum value.  Note that the ASCII driver also pauses between two telegrams during transmission.



## 7.22.6 RF180C

The Openness parameters for the "RF180C" module are described below.

Table 7-14 Parameters of the module "RF180C"

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
User mode	UserMode	ulong	FB 45	1	R/W	Selection depends on the communications module and Ident system being used. With this parameter, you select the block: <ul style="list-style-type: none"> <li>• FB 45: Single tag mode FB 45 (PROFIBUS/PROFINET) is used in the controller.</li> <li>• FB 55: Multitag mode. FB 55 (PROFIBUS/PROFINET) is used on the controller.</li> <li>• FB 56: Multitag mode. FB 56 (PROFIBUS/PROFINET) is used on the controller.</li> <li>• RFID standard profile The program block for the RFID standard profile is used on the controller.</li> </ul>
			FB 55	4	R/W	
			FB 56	5	R/W	
			RFID standard profile	3	R/W	
MOBY mode	MobyMode	ulong	MOBY I/E normal addressing	1	R/W	Selection depends on the communications module and Ident system being used. With this parameter, you set the mode of the communications module. <ul style="list-style-type: none"> <li>• MOBY I/E normal addressing</li> <li>• MOBY I file handler</li> <li>• RF200/RF300/RF600; MOBY U/D normal addr.</li> <li>• MOBY U file handler</li> <li>• RF300 filehandler</li> </ul> Normal addressing: The transponder is addressed with physical addresses. Filehandler: Prior to use, the transponder needs to be formatted.
			MOBY I file handler	129	R/W	
			RF200/RF300/RF600; MV4x0; MOBY U/D normal addr.	5	R/W	
			MOBY U file handler	133	R/W	
			RF300 filehandler	149	R/W	

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set the transmission speed between the communications module and reader.</p> <p>When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).</p> <p>When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.</p>
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	
Diagnostics messages	Diagnostic-Messages	int	None	1	R/W	<p>With this parameter, you set whether hardware diagnostics messages will be reported.</p> <ul style="list-style-type: none"> <li>• None: Apart from standard diagnostics, no other alarms are generated.</li> <li>• Hard errors: Critical hardware errors are reported by the S7 diagnostics.</li> <li>• Hard/Soft Errors: Critical hardware faults and errors occurring when processing commands are reported by the S7 diagnostics.</li> </ul>
			Hard errors	2	R/W	
			Hard/soft errors	4	R/W	
Suppression of Error LED	SuppressionOfErrorLed	int	None	0	R/W	<p>Disabling the Error LED (ERR) of a channel.</p> <p>The communications module has two channels to which the readers / optical readers can be connected. The Error LED of the other channel flashes permanently when only one of the channels is being used. With the help of the suppression, you can disable the Error LED of the unused channel.</p>
			Channel 1	1	R/W	
			Channel 2	2	R/W	

### 7.22.7 RF18xC

The Openness parameters for the "RF18xC" module are described below.

Table 7-15 Parameters of the modules "RF18xC"; parameters of the parameter group "Basic parameter"

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
User mode	UserMode	ulong	Ident profile/ RFID standard profile	3	R/W	Selection depends on the communications module and Ident system being used. With this parameter, you select the block: <ul style="list-style-type: none"> <li>Ident profile/RFID standard profile: The program block for the Ident profile/RFID standard profile is used on the controller.</li> </ul>

Table 7-16 Parameters of the "RF18xC" modules; parameters of the parameter group "Module parameters"

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
User mode	UserMode	ulong	FB 45	1	R/W	Selection depends on the communications module and Ident system being used. With this parameter, you select the block: <ul style="list-style-type: none"> <li>FB 45: Single tag mode FB 45 (PROFIBUS/PROFINET) is used in the controller.</li> <li>Ident profile/RFID standard profile: The program block for the Ident profile/RFID standard profile is used on the controller.</li> </ul>
			Ident profile/ RFID standard profile	3	R/W	

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
MOBY mode	MobyMode	ulong	RF200/ RF300/ RF600; MV4x0; MO- BY U/D nor- mal add.	5	R	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set the mode of the communications module.</p> <ul style="list-style-type: none"> <li>RF200/RF300/RF600; MOBY U/D normal addr.</li> </ul> <p>Normal addressing: The transponder is addressed with physical addresses.</p> <p>Filehandler: Prior to use, the transponder needs to be formatted.</p>
Transmission speed	BaudRate	ulong	19.2 kBd	9	R/W	<p>Selection depends on the communications module and Ident system being used. With this parameter, you set the transmission speed between the communications module and reader.</p> <p>When the RFID reader is connected: After changing the transmission speed, the reader must be turned off and on again (cycle power).</p> <p>When an optical reader is connected: The transmission speed selected here must match the transmission speed selected in the firmware of the reader.</p>
			57.6 kBd	13	R/W	
			115.2 kBd	14	R/W	

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
Diagnostics messages	Diagnostic-Messages	int	None	1	R/W	With this parameter, you set whether hardware diagnostics messages will be reported. <ul style="list-style-type: none"> <li>• None: Apart from standard diagnostics, no other alarms are generated.</li> <li>• Hard errors: Critical hardware errors are reported by the S7 diagnostics.</li> <li>• Hard/Soft Errors: Critical hardware faults and errors occurring when processing commands are reported by the S7 diagnostics.</li> </ul>
			Hard errors	2	R/W	
			Hard/soft errors	4	R/W	

### 7.22.8 RF615R/RF680R/RF685R

The Openness parameters for the "RF615R/RF680R/RF685R" modules are described below.

Table 7-17 Parameters of the modules "RF615R/RF680R/RF685R"

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
RFID read points alarm 1	RfidRead-PointAlarm1	int	Off	0	R/W	Enabling/disabling read point-related diagnostics messages.
			On	1	R/W	
RFID read points alarm 2	RfidRead-PointAlarm2	int	Off	0	R/W	
			On	1	R/W	
RFID read points alarm 3 <sup>1)</sup>	RfidRead-PointAlarm3	int	Off	0	R/W	
			On	1	R/W	
RFID read points alarm 4 <sup>1)</sup>	RfidRead-PointAlarm4	int	Off	0	R/W	
			On	1	R/W	

<sup>1)</sup> Only with RF680R

### 7.22.9 MV400/MV500

The Openness parameters for the "MV400/MV500" modules are described below.

Table 7-18 Parameters of the modules "MV400/MV500"

Parameters in the configuration (TIA Portal)	Parameters in Openness	Data type in Openness	Parameter value in the configuration (TIA Portal)	Parameter value in Openness	Default access	Description
Function block	FunctionBlock	int	FB79	0	R/W	With this parameter, you select the block: <ul style="list-style-type: none"> <li>• FB79 Compatible with the "VS130-2" reader; Compatible controllers: S7-300 and S7-400</li> <li>• Ident profile Complex Ident block with MV-specific blocks; Compatible controllers: S7-300, S7-400, S7-1200 and S7-1500</li> </ul>
			Ident profile	1	R/W	

## **7.23 Exceptions**

### **7.23.1 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);
}
```



# Export/import

## 8.1 Overview

### 8.1.1 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 8-1 Projects

Objects	Export	Import
Project graphics	X	X

Table 8-2 PLC

Objects	Export	Import
Blocks	X	X
Know-how protected blocks	X	–
Failsafe blocks	X	–
System blocks	X	–
PLC tag tables	X	X
PLC tags and constants	X	X
User data types	X	X

Table 8-3 HMI

Objects	Export	Import
Screens	X	X
Screen templates	X	X
Global screens	X	X
Pop-up screens	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 417)).

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 425)
- XML format for the data of a PLC devices (Page 475)
- AML format for CAx data (Page 546)

## 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.

## 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 407).

## See also

Field of application for Import/Export (Page 413)

Exporting configuration data (Page 415)

## 8.1.2 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 411)

### 8.1.3 Version Specific Simatic ML Import

#### Application

As of TIA Portal Openness V14 SP1 the SimaticML import is useable cross-version. You will be able to import your older export files at least into the next two major versions.

Each version of the Openness API is able to import Simatic ML files from corresponding version and any supported version from previous release, for example import of Simatic ML file V14 SP1 will be supported in Openness API V15.1.

The following table shows an example of which Simatic ML version can be imported by a given Openness API version.

	<b>Simatic ML file V14 SP1</b>	<b>Simatic ML file V15</b>	<b>Simatic ML V15.1</b>
Openness API V14 SP1	Import supported	Import unsupported	Import unsupported
Openness API V15 SP1	Import supported	Import supported	Import unsupported
Openness API V15.1	Import supported	Import supported	Import supported

Each version of the Openness API supports export of Simatic ML files. However, the version of the exported Simatic ML file should match with the version of the TIA Portal rather than that of the Openness API used.

To support this feature, SimaticML files contains the model version information as shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1"/>
  <DocumentInfo>
    ...
  </DocumentInfo>
  <SW.DataBlock ID="0">
    ...
  </SW.DataBlock>
</Document>
```

---

**Note**

If the version information is not provided in the SimaticML file, the system will use the current model version.

---

## 8.1.4 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.

---

## 8.1.5 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`<sup>1</sup>  
The default values are also exported.
- `ExportOptions.WithReadOnly`<sup>1</sup>  
The write-protected values are also exported.

<sup>1</sup>: 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 411)

Exporting blocks (Page 486)



## 8.1.6 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.
- 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.
- The import of failsafe blocks is not supported.

---

### 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 overwriting. 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 overwriting. 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.

### 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 407)).

### See also

Basic principles of importing/exporting (Page 411)

Importing user data type (Page 543)

## 8.2 Import/export of project data

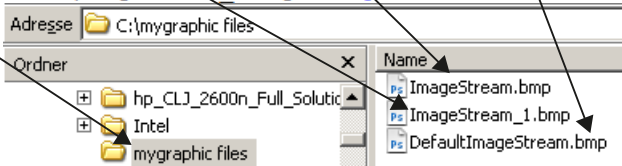
### 8.2.1 Project graphics

#### 8.2.1.1 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 411)

### 8.2.1.2 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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 graphic
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);
}
```

## 8.2.1.3 Importing graphics to a project

### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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 417).

### 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);
```

## 8.2.2 Project texts

### 8.2.2.1 Export of project texts

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

#### 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>
...
```

**8.2.2.2 Import of project texts****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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.
- 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
updateSourceLanguage	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\nProjectText.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;
```



## 8.3 Importing/exporting data of an HMI device

### 8.3.1 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.

```

<?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>

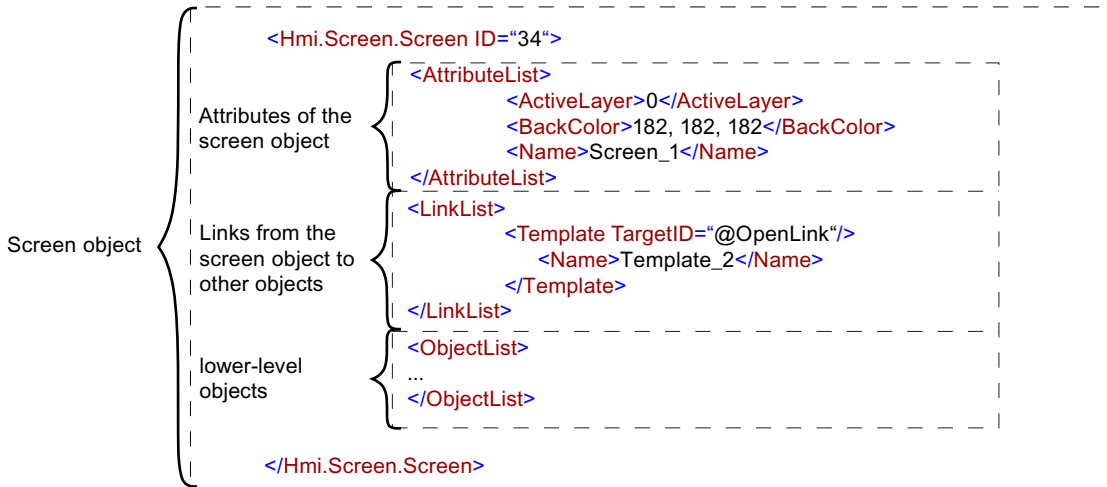
```

Information about the document

Screen object

## 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 411)

## 8.3.2 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.

```

<Hmi.Screen.Screen ID="34">
  <AttributeList>
    <ActiveLayer>0</ActiveLayer>
    <BackColor>182, 182, 182</BackColor>
    <Name>Screen_1</Name>
  </AttributeList>
  <LinkList>
    <Template TargetID="@OpenLink">
      <Name>Template_2</Name>
    </Template>
  </LinkList>
  <ObjectList>
    <Hmi.Screen.ScreenLayer ID="1" CompositionName="Layers">
      <AttributeList>
        <Index>0</Index>
        <Name>Layer_0</Name>
      </AttributeList>
      <ObjectList>

```

Figure 8-1 Relation between the WinCC user interface and the XML structure.

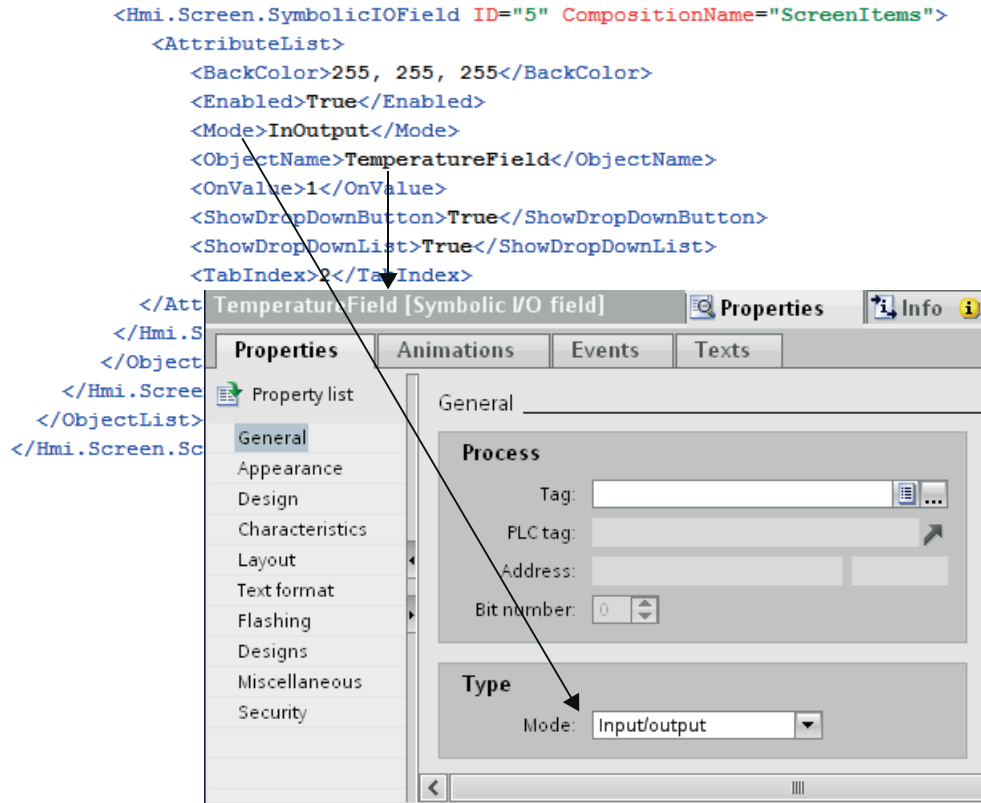


Figure 8-2 Relation between the settings in WinCC and the XML structure.

### 8.3.3 Cycles

#### 8.3.3.1 Exporting cycles

##### Requirements

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

##### 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);
    }
}
```

### 8.3.3.2 Importing cycles

#### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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 fullPath = Path.Combine(dirPathImport, cycleImportFileName);

    cycles.Import(new FileInfo(fullPath), ImportOptions.None);
}
```

## See also

Importing configuration data (Page 417)

## 8.3.4 Tag tables

### 8.3.4.1 Exporting HMI tag tables

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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 fullPath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(new FileInfo(fullPath), ExportOptions.WithDefaults);
    }
}

private static void ExportTablesInSystemFolder(TagSystemFolder folderToExport)
{
    TagTableComposition tables = folderToExport.TagTables;
    foreach (TagTable table in tables)
    {
        string fullPath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(new FileInfo(fullPath), ExportOptions.WithDefaults);
    }
}
```

### 8.3.4.2 Importing HMI tag table

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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>
```

### 8.3.4.3 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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);
    }
}
```

### 8.3.4.4 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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);
}
```

### 8.3.4.5 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 436)
- Program code: Importing HMI tags (Page 437)

**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

**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>
```

**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>
```

## 8.3.5 VB scripts

### 8.3.5.1 Exporting VB scripts

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### Application

All sublevel user-defined folders are taken into account for the export. A separate XML file is created for each exported VB script.

### 8.3 Importing/exporting data of an HMI device

#### 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);
}
```

#### 8.3.5.2 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 74)
- A project is open.  
See Opening a project (Page 97)

##### 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);
    }
}
```

**8.3.5.3 Importing VB scripts****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## Application

Bulk imports are supported. As an alternative, you can use a program code with a Foreach loop (Exporting VB scripts (Page 439)).

## Program code

Modify the following program code to import a VB script from an XML file into an HMI device:

```
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);
    }
}
```

## 8.3.6 Text lists

### 8.3.6.1 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### Application

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:

```
//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);
    }
}
```

### 8.3.6.2 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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);
}
```

### 8.3.6.3 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 74)
- A project is open.  
See Opening a project (Page 97)
- Standard export of text lists  
See Exporting text lists from an HMI device (Page 442)
- Standard import of text lists  
See Importing text lists into an HMI device (Page 443)

#### 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_li
                </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_li
                </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>

```

## 8.3.7 Graphic lists

### 8.3.7.1 Exporting graphic lists

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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);
    }
}
```

### 8.3.7.2 Importing a graphic list

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## 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);
}
```

## 8.3.8 Connections

### 8.3.8.1 Exporting connections

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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));
        connection.Export(info, ExportOptions.WithDefaults);
    }
}
```

#### 8.3.8.2 Importing connections

##### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

##### 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);
}
```

## 8.3.9 Screens

### 8.3.9.1 Overview of exportable screen objects

#### Application

You can export or import the screens below using TIA Portal Openness APIs:

Table 8-4 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 8-5 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 411)

### 8.3.9.2 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 74)
- A project is open.  
See Opening a project (Page 97)

#### Application

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:

```
private static void ExportScreensOfDevice(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);
}
```

#### Program code: Exporting all screens of a user-definded folder

Modify the following program code to 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);
    }
}
```

**Program code: Exporting all screens of a device independent of the user**

Modify the following program code to export all screens:

```
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);
    }
}

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);
    }
}
```

**8.3.9.3 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 74)
- A project is open.  
See Opening a project (Page 97)

## Application

The following data of a screen is exported:

Screen	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Open links	Template
Compositions	<ul style="list-style-type: none"> <li>• Layers</li> <li>• Animations All configured animations that are based on Runtime Advanced are exported.</li> <li>• Events All configured events that are based on Runtime Advanced are exported.</li> <li>• Softkeys All configured softkeys are exported.</li> </ul>

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);
    }
}
```

#### 8.3.9.4 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 74)
- A project is open. See Opening a project (Page 97)

##### 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	<ul style="list-style-type: none"><li>• Layers</li><li>• Animations All animations configurable for screens are imported.</li><li>• Events All events configurable for screens are imported.</li><li>• Softkeys All softkeys configurable for screens are imported.</li></ul>



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.

---

8.3 Importing/exporting data of an HMI device

**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);
}
```

**8.3.9.5 Exporting permanent areas**

**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

**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);
}
```

### 8.3.9.6 Importing permanent areas

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### 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.

### 8.3 Importing/exporting data of an HMI device

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);
}
```

#### 8.3.9.7 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 \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

##### Application

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:

```
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);
}
```

**Program code: Exporting all screen templates of a specific folder**

Modify the following program code to export all screen templates:

```
//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);
    }
}
```

**8.3.9.8 Exporting screen templates from a folder****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

**Application**

The following data of the screen template is exported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name
Compositions	<ul style="list-style-type: none"> <li>• Layers</li> <li>• Animations All configured animations are exported. SCADA animations are not exported.</li> <li>• Softkeys All configured softkeys are exported.</li> </ul>

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:

```
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);
}
```

**Program code: Exporting all screen templates of a user-defined folder**

Modify the following program code to export all screen templates from a specific 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);
    }
}
```

**Program code: Exporting all screen templates of a specific folder**

Modify the following program code to export all screen templates:

```
//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 folders.Folders)
    {
        ExportScreenTemplates(Path.Combine(templatePath, subfolder.Name), subfolder);
    }
}
```

**8.3.9.9 Importing screen templates****Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

**Application**

The following data of a screen template is imported:

Screen template	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, SetTabOrderInFront
Compositions	<ul style="list-style-type: none"> <li>• Layers</li> <li>• Animations All animations configurable for screens are imported.</li> <li>• Softkeys All softkeys configurable for screens are imported.</li> </ul>

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:

```
//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);
    }
}
```

### Program code: Import into a newly created user folder

Modify the following program code to import a screen template into a newly created user folder of an HMI device:

```
//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);
}
```

## 8.3.9.10 Exporting a pop-up screen

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)



## Application

The following data of the pop-up screen is exported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
Compositions	<ul style="list-style-type: none"> <li>• Layers</li> <li>• Events</li> </ul> All configured events are exported.

The following data is exported for each layer:

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems All exportable 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:

```
//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);
}
```

#### 8.3.9.11 Importing a pop-up screen

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## Application

The following data of a pop-up screen is imported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
Compositions	<ul style="list-style-type: none"> <li>• Layers</li> <li>• Events</li> </ul> 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);
}
```

### 8.3.9.12 Exporting a slide-in screen

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)

#### 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	
Attributes	Name,	
	Index	
	VisibleES	
Compositions	ScreenItems	All exportable 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);
}
```

**8.3.9.13 Importing a slide-in screen**

**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

**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);
}
```

### 8.3.9.14 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 74)
- A project is open.  
See Opening a project (Page 97)

## 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	<ul style="list-style-type: none"> <li>• Animations All movement animations are exported. Tag animations rely on interface attributes.</li> <li>• Events All configured events are exported.</li> </ul>

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);
    }
}
```

### 8.3.9.15 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 74)
- A project is open.  
See Opening a project (Page 97)

#### 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	<ul style="list-style-type: none"> <li>• Animations All movement animations are imported. Tag animations rely on interface attributes.</li> <li>• Events All configured events are imported.</li> </ul>

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" und "MinimumValue" are configured al 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);
}
```

## 8.4 Importing/exporting data of a PLC device

### 8.4.1 Blocks

#### 8.4.1.1 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="&quot;User_data_type_1&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 Re- adOnly (informa- tive)	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 assignment at call if matches with PredefinedAssignment
PredefinedAssignment	string	""	FALSE	Input for the parameter used when call is placed
ReadOnlyAssignment	bool	FALSE	FALSE	The user cannot change the predefined assignment at the call
UserVisible	bool	TRUE	TRUE	This member is not shown on the UI

Name	Type	Default	SimaticML Re- adOnly (informa- tive)	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 Re- adOnly (informa- tive)	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="&quot;User_data_type_1&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 &quot;User_data_type_1&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.

### 8.4.1.2 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 V14\PublicAPI\V14 SP1\Schemas  
\SW.InterfaceSections_v2.xsd
```

## Compile unit

There are separate schemas for the compile units of GRAPH, LAD/FBD and STL blocks. You can find these schemas in the following directories:

- GRAPH: C:\Program Files\Siemens\Automation\Portal V ...\PublicAPI\Schemas\SW.PlcBlocks.Graph.xsd
- LAD/FBD: C:\Program Files\Siemens\Automation\Portal V ...\PublicAPI\Schemas\SW.PlcBlocks.LADFB.D.xsd
- STL: C:\Program Files\Siemens\Automation\Portal V ...\PublicAPI\Schemas\SW.PlcBlocks.STL.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 V14\PublicAPI\W ...\Schemas\SW.PlcBlocks.Access.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 V14\PublicAPI\W ...\Schemas\SW.Common.xsd

### 8.4.1.3 Exporting blocks

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- PLC is not online.

## Application

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 415)). 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 <sup>1</sup>	Bool	false	true
IsWriteProtected	Bool	false	true
MemoryLayout	enum MemoryLayout	-	false
ModifiedDate	DateTime	-	true
<b>Name</b>	String	-	false
Number	Int32	next available number	false
ParameterModified	DateTime	-	true

Attribute	Type	Default value	ReadOnly
PLCSimAdvancedSupport	Bool	false	true
ProgrammingLanguage	enum ProgrammingLanguage	-	false
StructureModified	DateTime	-	true

<sup>1</sup> The IsKnowHowProtected attribute is applicable for UDT too.

### 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 <sup>1</sup>	Bool	false	false
MemoryReserve	Unsigned	0	false
RetainMemoryReserve <sup>2</sup>	Unsigned	0	false

<sup>2</sup> 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
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	OBDDataExchangeMode	Cyclic	true
DelayTime	Double	-	true
DistributedIOName	String	-	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 <sup>3</sup>	String	-	false
StartDate	DateTime	1/1/2012	true
SynchronousApplicationCycleTime	Single	-	true
TimeMode	OBTimeMode	System	true

Attribute	Type	Default value	ReadOnly
TimeOfDay	DateTime	12:00 AM	true
TransformationDBNumber	UInt16	0xffff	true

- <sup>3</sup> 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 attributes 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

### 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);
}
```

#### 8.4.1.4 Exporting DBs with snapshots

##### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)

## Application

You can use the TIA Portal Openness to export the DB's with snapshot values as XML and able to compare the values with different snapshot times. With the compare result, you can manually adapt single start values within the UI and store for a potential recovery.

## Program code

Modify the following program code to export Snapshot Values by using the Snapshot Service:

```
InterfaceSnapshot interfaceSnapshot = dataBlock.GetService<InterfaceSnapshot>();  
interfaceSnapshot.Export(new FileInfo("C:\\temp\\MyInterfaceSnapshot.xml"),  
ExportOptions.None);
```

The Snapshot Service "InterfaceSnapshot" will be provided in the namespace "Siemens.Engineering.SW.Blocks". The file handling (e.g. if the export directory does not exist; creating of the export directory; if the export directory is readonly; if the export file already exists) will be the same as the standard interface openness export. The Snapshot Service will be supported for Global DBs, Instance DBs and Array DBs.

---

### Note

Export of snapshot values using the Snapshot Service is independent of the standard interface openness export and therefore 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>
        <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 would look like as following:

```
<SnapshotValues xmlns="http://www.siemens.com/automation/Openness/SW/Interface/Snapshot/v1"></SnapshotValues>
```

## See also

Opening a project (Page 97)

### 8.4.1.5 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 74)
- A project is open.  
See Opening a project (Page 97)
- PLC is not online.

#### Application

The resulting XML file is similar to the export file of a block without know-how protection. However, the export covers only the data of the user interface that is visible when the block is opened without a password.

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);
}
```

### 8.4.1.6 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 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 represented 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" />

### Indentation of SCL block statements

In TIA portal settings, you can modify the indentation of SCL code by accessing Options/Settings/General/Script/text editors. The following table defines the type of indentation based on the indent mode.

Ident mode	Result
None	Import operation adds the spaces as available in the source files.
Paragraph or Smart	Import operation adds the specified indent spaces in the imported file.

Based on the chosen indentation, the imported SCL block XML file are indented.

### 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 multiple 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 > <b>The nested comment is part of outer comment text.</b>
(* 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
<pre>region myregion ... end_region here is the end of myregion</pre>	<pre>&lt;Token Text="REGION" /&gt; &lt;Blank /&gt; &lt;Text&gt;myregion&lt;/Text&gt; &lt;NewLine /&gt; ... &lt;Token Text="END_REGION" /&gt; &lt;Blank /&gt; &lt;Text&gt;here is the end of myregion&lt;/ Text&gt; &lt;NewLine /&gt;</pre>
<pre>region // here are no blanks ... end_region</pre>	<pre>&lt;Token Text="REGION" /&gt; &lt;NewLine /&gt; &lt;LineComment .../&gt; &lt;Token Text="END_REGION" /&gt; &lt;NewLine /&gt;</pre>
<pre>region ... end_region;</pre>	<pre>&lt;Token Text="REGION" /&gt; &lt;NewLine /&gt; ... &lt;Token Text="END_REGION" /&gt; &lt;Text&gt;;&lt;/Text&gt; &lt;NewLine /&gt;</pre>

### 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
<pre>{PRAGMA_BEGIN 'Param1', 'Param2' (*parm 2*)} // something else {PRAGMA_END}</pre>	<pre>&lt;Token Text="{ " /&gt; &lt;Token Text="PRAGMA_BEGIN" /&gt; &lt;Blank /&gt; &lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;'Param1'&lt;/ ConstantValue&gt;   &lt;/Constant&gt; &lt;/Access&gt; &lt;Token Text="," /&gt; &lt;Blank /&gt; &lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;'Param2'&lt;/ ConstantValue&gt;   &lt;/Constant&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;LineComment Inserted="True"&gt;   &lt;Text&gt;param 2&lt;/Text&gt; &lt;/LineComment&gt; &lt;Token Text="," /&gt; &lt;Blank /&gt; &lt;Token Text="}" /&gt; &lt;NewLine /&gt; &lt;LineComment&gt;   &lt;Text&gt; something else&lt;/Text&gt; &lt;/LineComment&gt; &lt;NewLine /&gt; &lt;Token Text="{ " /&gt; &lt;Token Text="PRAGMA_END" /&gt; &lt;Token Text="}" /&gt;</pre>

### 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 preceded 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;	<pre>&lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;10&lt;/ConstantValue&gt;     &lt;ConstantTypeInformative="true"&gt;LINT&lt;/ ConstantType&gt;   &lt;/Constant&gt; &lt;/Access&gt;</pre>
Literal constant: String	#myString := 'Hello world';	<pre>&lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;Hello world&lt;/ ConstantValue&gt;  &lt;ConstantTypeInformative="true"&gt;STRING&lt;/ ConstantType&gt;   &lt;/Constant&gt; &lt;/Access&gt;</pre>
Literal constant: Typed	#Out := int#10;	<pre>&lt;Access Scope="TypedConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;int#10&lt;/ConstantValue&gt;   &lt;/Constant&gt; &lt;/Access&gt;</pre> <p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre>&lt;Access Scope="TypedConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;int#10&lt;/ConstantValue&gt;     &lt;StringAttribute Name="Format" Informative="true"&gt;Dec_signed&lt;/ StringAttribute&gt;     &lt;StringAttribute Name="FormatFlags" Informative="true"&gt;TypeQualifier&lt;/ StringAttribute&gt;   &lt;/Constant&gt; &lt;/Access&gt;</pre>
Local constant	#Out := #mylocal;	<pre>&lt;Access Scope="LocalConstant"&gt;   &lt;Constant Name="mylocal" /&gt; &lt;/Access&gt;</pre> <p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre>&lt;Access Scope="LocalConstant"&gt;   &lt;Constant Name="mylocal"&gt;     &lt;ConstantType Informative="true"&gt;Int&lt;/ ConstantType&gt;     &lt;ConstantValue Informative="true"&gt;10&lt;/ ConstantValue&gt;     &lt;StringAttribute Name="Format" Informative="true"&gt;Dec_signed&lt;/ StringAttribute&gt;   &lt;/Constant&gt; &lt;/Access&gt;</pre>

Type of constant	SCL block	XML tag
Global constant	#Out := "myglobal";	<pre>&lt;Access Scope="GlobalConstant"&gt;   &lt;Constant Name="myglobal" /&gt; &lt;/Access&gt;</pre>
		<p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre>&lt;Access Scope="GlobalConstant"&gt;   &lt;Constant Name="myglobal"&gt;     &lt;ConstantType Informative="true"&gt;Int&lt;/ ConstantType&gt;     &lt;ConstantValue Informative="true"&gt;10&lt;/ ConstantValue&gt;     &lt;StringAttribute Name="Format" Informative="true"&gt;Dec_signed&lt;/ StringAttribute&gt;   &lt;/Constant&gt; &lt;/Access&gt;</pre>

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;	<pre>&lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="Out" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>
Global variable	"Tag_3" := 10;	<pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="Tag_3" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>
		<p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="Tag_3" /&gt;     &lt;Address Area="Memory" Type="Int" BitOffset="96" Informative="true" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>

## 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;	<pre> &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="a" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token text=":=" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="b" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token text="+" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="c" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Token text=";" /&gt; </pre>

### Control structures in SCL blocks

The control statements like IF, CASE, FOR, WHILE, REPEAT, GOTO, EXIT, CONTINUE, and RETURN are represented by Token XML tag.

- The conditional symbols used in SCL block such as >, <, & are represented as escape sequences (&lt; &gt; &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	<pre>IF #a&lt;#c THEN ; END_IF;</pre>	<pre>&lt;Token Text="IF" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="a" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Token Text="&amp;lt;" /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="c" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text="THEN" /&gt; &lt;NewLine /&gt; &lt;Blank Num="4" /&gt; &lt;Token Text=";" /&gt; &lt;NewLine /&gt; &lt;Token Text="END_IF" /&gt; &lt;Token Text=";" /&gt;</pre>

8.4 Importing/exporting data of a PLC device

Name of the block	SCL block	XML tag
CASE	<pre> CASE #a OF   1 (*test*): // Statement section case 1   ;   2..4: // Statement section case 2 to 4   ;   ELSE // Statement section ELSE   ; END_CASE; </pre>	<pre> &lt;Tok en Text="CASE" /&gt;&lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="a" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text="OF" /&gt; &lt;NewLine /&gt;  &lt;Blank Num="2"/&gt; &lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;1&lt;/ConstantValue&gt;     &lt;ConstantType Informative="true"&gt;LINT&lt;/ConstantType&gt;   &lt;/Constant&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;LineComment Inserted="true"&gt;   &lt;Text&gt;test&lt;/Text&gt; &lt;/LineComment &gt; &lt;Token Text=":" /&gt; &lt;Blank /&gt; &lt;LineComment&gt;   &lt;Text&gt; Statement section case 1&lt;/Text&gt; &lt;/LineComment &gt; &lt;NewLine /&gt;  &lt;Blank Num="4"/&gt; &lt;Token Text=";" /&gt; &lt;NewLine /&gt;  &lt;Blank Num="2"/&gt; &lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;2&lt;/ConstantValue&gt;     &lt;ConstantType Informative="true"&gt;LINT&lt;/ConstantType&gt;   &lt;/Constant&gt; &lt;/Access&gt; &lt;Token Text=".." /&gt; &lt;Blank Num="2"/&gt; &lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;     &lt;ConstantValue&gt;4&lt;/ConstantValue&gt;     &lt;ConstantType Informative="true"&gt;LINT&lt;/ConstantType&gt;   &lt;/Constant&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;LineComment&gt;   &lt;Text&gt; Statement section case 2 to 4&lt;/Text&gt; &lt;/LineComment &gt; &lt;NewLine /&gt; </pre>



Name of the block	SCL block	XML tag
		<pre> &lt;Blank Num="4" /&gt; &lt;Token Text=";" /&gt; &lt;NewLine /&gt;  &lt;Blank Num="2" /&gt; &lt;Token Text="ELSE" /&gt; &lt;NewLine /&gt;  &lt;Blank Num="4" /&gt; &lt;Token Text=";" /&gt; &lt;NewLine /&gt;  &lt;Token Text="END_CASE" /&gt; &lt;Token Text=";" /&gt; </pre>
FOR	<pre> FOR #i := #a TO #b DO   // Statement section FOR   ; END_FOR; </pre>	<pre> &lt;Token Text="FOR" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="i" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text=":=" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="a" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text="TO" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="b" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text="DO" /&gt; &lt;NewLine /&gt;  &lt;Blank Num="2" /&gt; &lt;LineComment&gt;   &lt;Text&gt; Statement section FOR&lt;/Text&gt; &lt;/LineComment &gt; &lt;NewLine /&gt;  &lt;Blank Num="2" /&gt; &lt;Token Text=";" /&gt; &lt;NewLine /&gt;  &lt;Token Text="END_FOR" /&gt; &lt;Token Text=";" /&gt; </pre>

Name of the block	SCL block	XML tag
WHILE	<pre> WHILE #a&lt;#b DO   // Statement section WHILE   ; END_WHILE;                     </pre>	<pre> &lt;Token Text="WHILE" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="a" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Token Text="&amp;lt;" /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="b" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text="DO" /&gt; &lt;NewLine /&gt;  &lt;Blank Num="2" /&gt; &lt;LineComment&gt;   &lt;Text&gt; Statement section WHILE&lt;/Text&gt; &lt;/LineComment &gt; &lt;NewLine /&gt;  &lt;Blank Num="2" /&gt; &lt;Token Text=";" /&gt; &lt;NewLine /&gt;  &lt;Token Text="END_WHILE" /&gt; &lt;Token Text=";" /&gt;                     </pre>

Name of the block	SCL block	XML tag
REPEAT	<pre> REPEAT   // Statement section REPEAT   ; UNTIL #a&lt;#b END_REPEAT;</pre>	<pre> &lt;Token Text="REPEAT" /&gt; &lt;NewLine /&gt;  &lt;Blank Num="2" /&gt; &lt;LineComment&gt;   &lt;Text&gt; Statement section REPEAT&lt;/Text&gt; &lt;/LineComment &gt; &lt;NewLine /&gt;  &lt;Blank Num="2" /&gt; &lt;Token Text=";" /&gt; &lt;NewLine /&gt;  &lt;Token Text="UNTIL" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="a" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Token Text="&amp;lt;" /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="b" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text="END_REPEAT" /&gt; &lt;Token Text=";" /&gt;</pre>

Name of the block	SCL block	XML tag
GOTO	<pre>                     here                     // well                     : // this is goto statement                 </pre>	<p>XML example for GOTO label definition</p> <pre> &lt;Blank Num="3"/&gt; &lt;Access Scope="Label"&gt;   &lt;Label Name="here"&gt;     &lt;NewLine /&gt;     &lt;Blank Num="3"/&gt;     &lt;LineComment&gt;       &lt;Text&gt; well&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;NewLine /&gt;     &lt;Token Text=":" /&gt;   &lt;Blank /&gt; &lt;/Label&gt; &lt;/Access&gt; &lt;LineComment&gt;   &lt;Text&gt; this is goto statement&lt;/Text&gt; &lt;/LineComment&gt;                 </pre>
	GOTO (*comment*) here;	<p>XML example for GOTO label usage</p> <pre> &lt;Token Text="GOTO" /&gt; &lt;Blank /&gt; &lt;LineComment inserted="true"&gt;   &lt;Text&gt;comment&lt;/Text&gt; &lt;/LineComment&gt; &lt;Blank /&gt; &lt;Access Scope="Label"&gt;   &lt;Label Name="here" /&gt; &lt;/Access&gt; &lt;Token Text=";" /&gt;                 </pre>

**Referencing attributes**

The SCL block referencing attributes are represented by AccessModifier attribute of Component tag.

- For simple referencing, AccessModifier has value as Reference.
- For array referencing, AccessModifier has value as ReferenceToArray.

SCL block	XML tag
RefToUDT^(*RefToUDT*).element	<pre>&lt;Symbol&gt;   &lt;Component Name="RefToUDT"   AccessModifier="Reference" /&gt;   &lt;Token Text="^" /&gt;   &lt;LineComment Inserted="True"&gt;     &lt;Text&gt;RefToUDT&lt;/Text&gt;   &lt;/LineComment&gt;   &lt;Token Text="." /&gt;   &lt;Component Name="element" /&gt; &lt;/Symbol&gt;</pre>
RefToArrayOfUDT^(*RefToArrayOfUDT*)[#i].element	<pre>&lt;Symbol&gt;   &lt;Component Name="RefToArrayOfUDT"   AccessModifier="ReferenceToArray" /&gt;   &lt;Token Text="^" /&gt;   &lt;LineComment Inserted="True"&gt;     &lt;Text&gt;RefToArrayOfUDT&lt;/Text&gt;   &lt;/LineComment&gt;   &lt;Token Text="[" /&gt;   &lt;Access Scope=LocalVariable&gt;     &lt;Symbol&gt;       &lt;Component Name="i" /&gt;     &lt;/Symbol&gt;   &lt;/Access&gt;   &lt;Token Text="]" /&gt; &lt;/Component&gt;   &lt;Token Text="." /&gt;   &lt;Component Name="element" /&gt; &lt;/Symbol&gt;</pre>

### 8.4.1.7 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 structured 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
<pre>"Data_block_1".(*comment 1*)Static_1(*comment 2*).Static_2</pre>	<pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="Data_block_1" /&gt;     &lt;Token Text="." /&gt;     &lt;LineComment Inserted="True"&gt;       &lt;Text&gt;comment 1&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;Component Name="Static_1" / &gt;     &lt;LineComment Inserted="True"&gt;       &lt;Text&gt;comment 2&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;Token Text="." /&gt;     &lt;Component Name="Static_2" / &gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>
<pre>"Data_block_1".Static_1 := 10</pre>	<p><b>Format of XML exported in ExportOptions.None setting.</b></p> <pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="Data_block_1" /&gt;     &lt;Token Text="." /&gt;     &lt;Component Name="Static_1" / &gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre> <p><b>Format of XML exported in ExportOptions.ReadOnly setting.</b></p> <pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="Data_block_1" /&gt;     &lt;Token Text="." /&gt;     &lt;Component Name="Static_1" / &gt;     &lt;Address Area="DB" Type="Word" BlockNumber="1" BitOffset="0" Informative="true" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>

## 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"	<pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="a" /&gt;     &lt;Token Text="." /&gt;     &lt;Component Name="b"&gt;       &lt;BooleanAttribute Name="HasHash"&gt;TRUE&lt;/ BooleanAttribute&gt;     &lt;/Component&gt;     &lt;Token Text="." /&gt;     &lt;Component Name="c"&gt;       &lt;BooleanAttribute Name="HasQuotes"&gt;TRUE&lt;/ BooleanAttribute&gt;     &lt;/Component&gt;     &lt;Token Text="." /&gt;     &lt;Component Name="d"&gt;       &lt;BooleanAttribute Name="HasQuotes"&gt;TRUE&lt;/ BooleanAttribute&gt;       &lt;BooleanAttribute Name="HasHash"&gt;TRUE&lt;/ BooleanAttribute&gt;     &lt;/Component&gt;   &lt;/Symbol&gt; &lt;/Access /&gt;</pre>

## 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	<pre> &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="a" /&gt;     &lt;Token Text="." /&gt;     &lt;Component Name="b" AccessModifier="Array" /&gt;     &lt;Token Text="[" /&gt;     &lt;Access Scope=LocalVariable&gt;       &lt;Symbol&gt;         &lt;Component Name="i" /&gt;       &lt;/Symbol&gt;     &lt;/Access&gt;     &lt;Token Text="+" /&gt;     &lt;Access Scope=LocalVariable&gt;       &lt;Symbol&gt;         &lt;Component Name="j" /&gt;       &lt;/Symbol&gt;     &lt;/Access&gt;     &lt;Token Text="," /&gt;     &lt;Access Scope=LocalVariable&gt;       &lt;Symbol&gt;         &lt;Component Name="k" /&gt;       &lt;/Symbol&gt;     &lt;/Access&gt;     &lt;Token Text="+" /&gt;     &lt;Access Scope=LocalVariable&gt;       &lt;Symbol&gt;         &lt;Component Name="l" /&gt;       &lt;/Symbol&gt;     &lt;/Access&gt;     &lt;Token Text="]" /&gt;   &lt;/Component&gt;   &lt;Token Text="." /&gt;   &lt;Component Name="c" /&gt; &lt;/Symbol&gt; &lt;/Access&gt; </pre>

### 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	<pre>&lt;Access Scope="Address"&gt;   &lt;Symbol&gt;     &lt;Address Area="DB" BlockNumber="20" /&gt;     &lt;Blank /&gt;     &lt;Token Text="." /&gt;     &lt;Blank /&gt;     &lt;Address Area="DB" BitOffset="80" Type="Word"/&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>
%DB20.DBX10.3 := true;	<p>The following XML is valid for all languages except SCL.</p> <pre>&lt;Access Scope="Address"&gt; &lt;Address Area="DB" BlockNumber="20" BitOffset="83" Type="Bool" /&gt; &lt;/Access&gt;</pre> <p>The following XML is valid for SCL.</p> <pre>&lt;Access Scope="Address"&gt;   &lt;Symbol&gt;     &lt;Address Area="DB" BlockNumber="20" /&gt;     &lt;Token Text="." /&gt;     &lt;Address Area="DB" BitOffset="83" Type="Bool"/&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>

### 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;	<pre>&lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="Input_DB_ANY" / &gt;     &lt;Token Name="." /&gt;     &lt;Address BitOffset="19" Type="Bool" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>

## 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	<pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="tag_1" /&gt;     &lt;LineComment Inserted="True"&gt;       &lt;Text&gt;1&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;Token Text="." /&gt;     &lt;LineComment Inserted="True"&gt;       &lt;Text&gt;2&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;Component Name="member"/&gt;     &lt;LineComment Inserted="True"&gt;       &lt;Text&gt;3&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;Token Text="." /&gt;     &lt;LineComment Inserted="True"&gt;       &lt;Text&gt;4&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;Token Text="%x1" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>

## Peripheral access

The peripheral access is represented by Token tag.

SCL Block	XML Tag
"tag_1"(*1*).(*2*)member:P	<pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="tag_1" /&gt;     &lt;LineComment Inserted="True"&gt;       &lt;Text&gt;1&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;Token Text="." /&gt;     &lt;LineComment Inserted="True"&gt;       &lt;Text&gt;2&lt;/Text&gt;     &lt;/LineComment&gt;     &lt;Component Name="member"/&gt;     &lt;Token Text=":P" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt;</pre>

## 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 <pre>if TypeOf( #inVariant ) = TO_SpeedAxis then ... end_if</pre>	<pre>&lt;Token text="" /&gt; &lt;/Blank&gt; &lt;Access Scope="SystemType"&gt;   &lt;DataType&gt;TO_SpeedAxis&lt;/DataType&gt; &lt;/Access&gt;</pre>
Example for user defined type <pre>if TypeOf( #inVariant ) = "aUserDefinedType" then ... end_if</pre>	<pre>&lt;Token text="" /&gt; &lt;/Blank&gt; &lt;Access Scope="UserType"&gt;   &lt;DataType&gt;aUserDefinedType&lt;/ DataType&gt; &lt;/Access&gt;</pre>

### 8.4.1.8 Export/Import of SCL call blocks

#### SCL call blocks with export XML tags

SCL call parameters are represented 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
<pre>#Callee_Instance(Input_1 := 5);</pre>	<p>Format of XML exported in ExportOptions.None setting</p> <pre>&lt;Access Scope="Call"&gt;   &lt;CallInfo BlockType="FB"&gt;     &lt;Instance Scope="LocalVariable"&gt;       &lt;Component Name="Callee_Instance" /&gt;     &lt;/Instance&gt;     &lt;Token text="(" /&gt;       &lt;Parameter Name="Input_1"&gt;         &lt;Blank /&gt;         &lt;Token text=":=" /&gt;         &lt;Blank /&gt;         &lt;Access Scope="LiteralConstant"&gt;           &lt;Constant&gt;             &lt;ConstantType&gt;Int&lt;/ ConstantType&gt;             &lt;ConstantValue&gt;5&lt;/ ConstantValue&gt;           &lt;/Constant&gt;         &lt;/Access&gt;       &lt;/Parameter&gt;     &lt;Token text=")" /&gt;   &lt;/CallInfo&gt; &lt;/Access&gt; &lt;Token text=";" /&gt;</pre>
	<p>Format of XML exported in ExportOptions.ReadOnly setting</p> <pre>&lt;Access Scope="Call"&gt;   &lt;CallInfo BlockType="FB"&gt;     &lt;IntegerAttribute Name="BlockNumber" Informative="true"&gt;1&lt;/ IntegerAttribute&gt;     &lt;DateAttribute Name="ParameterModifiedTS" Informative="true"&gt;2016-10-24T08:27: 34&lt;/DateAttribute&gt;     &lt;Instance Scope="LocalVariable"&gt;       &lt;Component Name="Callee_Instance" /&gt;     &lt;/Instance&gt;     &lt;Token text="(" /&gt;       &lt;Parameter Name="Input_1"&gt;         &lt;StringAttribute Name="InterfaceFlags" Informative="true"&gt;S7_Visible&lt;/ StringAttribute&gt;         &lt;Blank /&gt;         &lt;Token text=":=" /&gt;</pre>

SCL block	XML tag
	<pre data-bbox="901 278 1426 680">&lt;Blank /&gt; &lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;     &lt;ConstantType&gt;Int&lt;/ ConstantType&gt;     &lt;ConstantValue&gt;5&lt;/ ConstantValue&gt;   &lt;/Constant&gt; &lt;/Access&gt; &lt;/Parameter&gt; &lt;Token text=")" /&gt; &lt;/CallInfo&gt; &lt;/Access&gt; &lt;Token text=";" /&gt;</pre>

**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) ;	<pre> &lt;Access Scope="Call"&gt;   &lt;CallInfo Name="Block_4"   BlockType="FB"&gt;     &lt;Instance     Scope="GlobalVariable"&gt;       &lt;Component Name="Block_4_DB" /     &gt;       &lt;/Instance&gt;       &lt;Token text="(" /&gt;       &lt;Parameter Name="a"&gt;         &lt;Token text=":=" /&gt;         &lt;Access         Scope="LiteralConstant"&gt;           &lt;Constant&gt;             &lt;ConstantType&gt;Bool&lt;/             ConstantType&gt;             &lt;ConstantValue&gt;TRUE&lt;/             ConstantValue&gt;           &lt;/Constant&gt;         &lt;/Access&gt;       &lt;/Parameter&gt;       &lt;Token text="," /&gt;       &lt;Parameter Name="b"       Informative="true"/&gt;       &lt;Parameter Name="c" &gt;         &lt;Token text=":=" /&gt;         &lt;Access         Scope="LiteralConstant"&gt;           &lt;Constant&gt;             &lt;ConstantType&gt;Bool&lt;/             ConstantType&gt;             &lt;ConstantValue&gt;True&lt;/             ConstantValue&gt;           &lt;/Constant&gt;         &lt;/Access&gt;       &lt;/Parameter&gt;       &lt;Parameter Name="d"       Informative="true"/&gt;       &lt;Token text=")" /&gt;     &lt;/CallInfo&gt;   &lt;/Access&gt; </pre>

### 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
<pre>"Block_4_DB" (TRUE) ;</pre>	<pre>&lt;Access Scope="Call"&gt;   &lt;CallInfo Name="Block_4" BlockType="FB"&gt;   &lt;Instance Scope="GlobalVariable"&gt;   &lt;Component Name="Block_4_DB" / &gt;   &lt;/Instance&gt;   &lt;Token text="(" /&gt;   &lt;NamelessParameter&gt;   &lt;Access Scope="LiteralConstant"&gt;   &lt;Constant&gt;   &lt;ConstantType&gt;Bool&lt;/ ConstantType&gt;   &lt;ConstantValue&gt;TRUE&lt;/ ConstantValue&gt;   &lt;/Constant&gt;   &lt;/Access&gt;   &lt;/NamelessParameter&gt;   &lt;Token text=")" /&gt;   &lt;/CallInfo&gt; &lt;/Access&gt;</pre>



## Expression as actual parameter

SCL block	XML tag
#Callee_Instance(Input_1 := #a+3);	<pre> &lt;Access Scope="Call"&gt;   &lt;CallInfo BlockType="FB"&gt;     &lt;Instance       Scope="LocalVariable"&gt;         &lt;Component           Name="Callee_Instance" /&gt;         &lt;/Instance&gt;         &lt;Token text="(" /&gt;         &lt;Parameter Name="Input_1"&gt;           &lt;Blank /&gt;           &lt;Token text=":" /&gt;           &lt;Blank /&gt;           &lt;Access Scope="LocalVariable"&gt;             &lt;Symbol&gt;               &lt;Component Name="a" /&gt;             &lt;/Symbol &gt;           &lt;/Access&gt;           &lt;Token text="+" /&gt;           &lt;Access             Scope="LiteralConstant"&gt;               &lt;Constant&gt;                 &lt;ConstantType&gt;Int&lt;/                 ConstantType&gt;                 &lt;ConstantValue&gt;3&lt;/                 ConstantValue&gt;               &lt;/Constant&gt;             &lt;/Access&gt;           &lt;/Parameter&gt;           &lt;Token text=")" /&gt;         &lt;/CallInfo&gt;       &lt;/Access&gt;     &lt;Token text=";" /&gt; </pre>

Expression as actual parameter without formal paramter

SCL block	XML tag
#Callee_Instance(#a+3);	<pre> &lt;Access Scope="Call"&gt;   &lt;CallInfo BlockType="FB"&gt;     &lt;Instance Scope="LocalVariable"&gt;       &lt;Component Name="Callee_Instance" /&gt;     &lt;/Instance&gt;     &lt;Token text="(" /&gt;     &lt;NamelessParameter&gt;       &lt;Access Scope="LocalVariable"&gt;         &lt;Symbol&gt;           &lt;Component Name="a" /&gt;         &lt;/Symbol &gt;       &lt;/Access&gt;       &lt;Token text="+" /&gt;       &lt;Access Scope="LiteralConstant"&gt;         &lt;Constant&gt;           &lt;ConstantType&gt;Int&lt;/ ConstantType&gt;           &lt;ConstantValue&gt;3&lt;/ ConstantValue&gt;         &lt;/Constant&gt;       &lt;/Access&gt;     &lt;/NamelessParameter&gt;     &lt;Token text=")" /&gt;   &lt;/CallInfo&gt; &lt;/Access&gt; &lt;Token text=";" /&gt; </pre>

Function call

SCL block	XML tag
#myInt := "MyFunction"(Param_1 := 1, Param_2 := 15, Param_3 := TRUE);	<pre> &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="myInt" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token text=":=" /&gt; &lt;Blank /&gt; &lt;Access Scope="Call"&gt;   &lt;CallInfo Name="MyFunction" BlockType="FC"&gt;     &lt;Token text="(" /&gt;     &lt;Parameter Name="Param_1"&gt;       ... </pre>

## Absolute call

In SCL, the call can be initiated using absolute address of the DB. Due to absolute address, the Name attribute 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 (...);	<pre>&lt;Access Scope="Call"&gt;   &lt;CallInfo Name="" BlockType="FB"&gt;     &lt;Instance       Scope="GlobalVariable"&gt;         &lt;Address Area="DB"           BlockNumber="20" /&gt;         &lt;/Instance&gt;         &lt;Token text="( " /&gt;         &lt;Parameter&gt;           ...         &lt;/Parameter&gt;         &lt;Token text=")" /&gt;       &lt;/CallInfo&gt;     &lt;/Access&gt;</pre>

## 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
<pre>#myInt := ATTACH(OB_NR := 1, EVENT := 15, ADD := TRUE);</pre>	<p>Format of XML exported in ExportOptions.ReadOnly setting</p> <pre>&lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="myInt" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token text=":" /&gt; &lt;Blank /&gt; &lt;Access Scope="Call"&gt;   &lt;Instruction Name="ATTACH"&gt;     &lt;Token text="(" /&gt;     &lt;Parameter Name="OB_NR"&gt;       &lt;Blank /&gt;       &lt;Token text=":" /&gt;       &lt;Blank /&gt;       &lt;Access Scope="LiteralConstant"&gt;         &lt;Constant&gt;           &lt;ConstantType&gt;OB_ATT&lt;/ ConstantType&gt;           &lt;ConstantValue&gt;1&lt;/ ConstantValue&gt;         &lt;/Constant&gt;       &lt;/Access&gt;     &lt;/Parameter&gt;     &lt;Token text="," /&gt;     &lt;Blank /&gt;     &lt;Parameter Name="EVENT"&gt;       &lt;Blank /&gt;       &lt;Token text=":" /&gt;       &lt;Blank /&gt;       &lt;Access Scope="LiteralConstant"&gt;         &lt;Constant&gt;           &lt;ConstantType&gt;EVENT_ATT&lt;/ ConstantType&gt;           &lt;ConstantValue&gt;15&lt;/ ConstantValue&gt;         &lt;/Constant&gt;       &lt;/Access&gt;     &lt;/Parameter&gt;     &lt;Token text="," /&gt;     &lt;Blank /&gt;     &lt;Parameter Name="ADD"&gt;       &lt;Blank /&gt;       &lt;Token text=":" /&gt;       &lt;Blank /&gt;       &lt;Access Scope="LiteralConstant"&gt;         &lt;Constant&gt;           &lt;ConstantType&gt;Bool&lt;/ ConstantType&gt;           &lt;ConstantValue&gt;TRUE&lt;/ ConstantValue&gt;         &lt;/Constant&gt;       &lt;/Access&gt;     &lt;/Parameter&gt;   &lt;/Instruction&gt; &lt;/Access&gt;</pre>

SCL block	XML tag
	<pre data-bbox="901 272 1308 534">&lt;/Constant&gt; &lt;/Access&gt; &lt;/Parameter&gt; &lt;Parameter Name="RET_VAL" Informative="true" /&gt;   &lt;Token text=")" /&gt; &lt;/Instruction&gt; &lt;/Access&gt; &lt;Token text=";" /&gt;</pre>

**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
<pre>"tag_4" := MIN_DINT( IN1:="Tag_1", IN2:="Tag_2", IN3:="Tag_3" );</pre>	<pre>&lt;Access Scope="GlobalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="Tag_4" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; ... &lt;Access Scope="Call"&gt;   &lt;Instruction Name="MIN"&gt;     &lt;TemplateValue Name="value_type" Type="Type"&gt;DInt&lt;/ TemplateValue&gt;     ...     &lt;Parameter Name="IN1"&gt;       ...       &lt;Access Scope="GlobalVariable"&gt;         &lt;Symbol&gt;           &lt;Component Name="Tag_1" /&gt;         &lt;/Symbol&gt;       &lt;/Access&gt;     &lt;/Parameter&gt;     ...     &lt;Parameter Name="IN2"&gt;...       &lt;Access Scope="GlobalVariable"&gt;         &lt;Symbol&gt;           &lt;Component Name="Tag_2" /&gt;         &lt;/Symbol&gt;       &lt;/Access&gt;     &lt;/Parameter&gt;     ...     &lt;Parameter Name="IN3"&gt;       ...       &lt;Access Scope="GlobalVariable"&gt;         &lt;Symbol&gt;           &lt;Component Name="Tag_3" /&gt;         &lt;/Symbol&gt;       &lt;/Access&gt;     &lt;/Parameter&gt;     ...   &lt;/Instruction&gt; &lt;/Access&gt; ...</pre>

## 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
<pre>#output_1 := TIME_TO_S5TIME(#input_1);</pre>	<pre>&lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="output_1" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; ... &lt;Access Scope="Call"&gt;   &lt;Instruction Name="TIME_TO_S5TIME"&gt;   &lt;Token text="( " /&gt;   &lt;NamelessParameter&gt;     &lt;Access Scope="LocalVariable"&gt;       &lt;Symbol&gt;         &lt;Component Name="input_1" /       &gt;     &lt;/Symbol&gt;   &lt;/Access&gt;   &lt;/NamelessParameter&gt;   &lt;Token text=")" /&gt; &lt;/Instruction&gt; &lt;/Access&gt; ...</pre>

### Instruction with instance

The instance and instruction are separated by blanks. Blanks are optional, and they can be represented by new lines and comments. The instruction TON is represented by Name attribute of Instruction tag.



SCL block	XML tag
<pre>IEC_Timer_0_DB . TON (IN:="Tag_1", PT:="Tag_2");</pre>	<pre>&lt;Access Scope="GlobalAccess"&gt;   &lt;Symbol&gt;     &lt;Component Name="IEC_Timer_0_DB" /&gt;   &lt;/Symbol &gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token text="." /&gt; &lt;Blank /&gt; &lt;Access Scope="Call"&gt;   &lt;Instruction Name="TON"&gt;     &lt;Blank /&gt;     &lt;Token text="(" /&gt;     &lt;Parameter Name="IN"&gt;       &lt;Access Scope="GlobalVariable"&gt;         &lt;Symbol&gt;           &lt;Component Name="Tag_1" /&gt;         &lt;/Symbol&gt;       &lt;/Access&gt;     &lt;/Parameter&gt;     ...     &lt;Token text=")" /&gt;   &lt;/Instruction&gt; &lt;/Access&gt; &lt;Token text=";" /&gt;</pre>

### 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
<pre>"Block_1_DB"(16#0000_0001);</pre>	<p>Format of XML exported in ExportOptions.None setting</p> <pre>&lt;Access Scope="Call"&gt;   &lt;CallInfo Name="Block_1" BlockType="FB"&gt;   &lt;Instance Scope="GlobalVariable"&gt;     &lt;Component Name="Block_1_DB" /&gt;   &lt;/Instance&gt;   &lt;NamelessParameter&gt;     &lt;Access Scope="AlarmConstant"&gt;       &lt;Constant&gt;         &lt;ConstantType&gt;C_Alarm_8&lt;/ ConstantType&gt;  &lt;ConstantValue&gt;16#0000_0001&lt;/ ConstantValue&gt;       &lt;/Constant&gt;     &lt;/Access&gt;   &lt;/NamelessParameter &gt; &lt;/CallInfo&gt; &lt;/Access&gt;</pre> <p>Format of XML exported in ExportOptions.ReadOnly setting</p> <pre>&lt;Access Scope="Call"&gt;   &lt;CallInfo Name="Block_1" BlockType="FB"&gt;   &lt;Instance Scope="GlobalVariable"&gt;     &lt;Component Name="Block_1_DB" /&gt;   &lt;/Instance&gt;   &lt;NamelessParameter&gt;     &lt;Access Scope="AlarmConstant" &gt;       &lt;Constant&gt;         &lt;ConstantValue&gt;16#00000001&lt;/ ConstantValue&gt;         &lt;ConstantType&gt;C_Alarm&lt;/ ConstantType&gt;         &lt;StringAttribute Name="Format" Informative="true"&gt;Hex&lt;/ StringAttribute&gt;       &lt;/Constant&gt;     &lt;/Access&gt;   &lt;/NamelessParameter&gt; &lt;/CallInfo&gt; &lt;/Access&gt;</pre>

## 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);	<pre>&lt;Access Scope="Call"&gt; &lt;CallInfo BlockType="FC"&gt;   &lt;Token text="("&gt; /&gt;   ...   &lt;Token text=","&gt; /&gt;   &lt;Blank /&gt;   &lt;Parameter Name="ENO"&gt;     &lt;Blank /&gt;     &lt;Token text="=&gt;" /&gt;     &lt;Blank /&gt;     &lt;Access       Scope="PredefinedVariable"&gt;         &lt;PredefinedVariable           Name="ENO" /&gt;         &lt;/Access&gt;       &lt;/Parameter&gt;     &lt;Token text=")"&gt; /&gt;   &lt;/CallInfo&gt; &lt;/Access&gt; &lt;Token text=";"&gt; /&gt;</pre>
IF ENO = #c THEN ...	<pre>&lt;Token text="IF" /&gt; &lt;Blank /&gt; &lt;Access Scope="PredefinedVariable"&gt;   &lt;PredefinedVariable Name="ENO" /&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text="=" /&gt; &lt;Blank /&gt; &lt;Access Scope="LocalVariable"&gt;   &lt;Symbol&gt;     &lt;Component Name="c" /&gt;   &lt;/Symbol&gt; &lt;/Access&gt; &lt;Blank /&gt; &lt;Token Text="THEN" /&gt;</pre>

### 8.4.1.9 Exporting failsafe blocks

#### Exporting failsafe blocks

Failsafe blocks are exported like standard blocks. For failsafe blocks the value of the attribute "ProgrammingLanguage" will start with a prefix "F\_".

---

#### Note

The import of a file is not possible if the value for attribute "ProgrammingLanguage" starts with a prefix "F\_".

---

#### Importing failsafe blocks as standard blocks

Failsafe blocks can be imported as standard blocks if the a prefix "F\_" is removed from the value of all attributes "ProgrammingLanguage".

#### See also

Connecting to the TIA Portal (Page 74)

Opening a project (Page 97)

Exporting blocks (Page 486)

Exporting blocks with know-how protection (Page 495)

### 8.4.1.10 Exporting system blocks

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- The project contains a system block.
- PLC is not online.

#### Application

Only visible system blocks will be available in the blocks composition, e.g. no SFBs or SFCs. 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);
        }
    }
}
```

### 8.4.1.11 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 respectively. 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>
```

#### 8.4.1.12 Importing block

##### Requirement

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)
- PLC is not online.

##### Application

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;
    IList<PlcBlock> blocks = systemblockGroup.Blocks.Import(new FileInfo(@"D:\Blocks
\myBlock.xml"), ImportOptions.Override);
}
```

### 8.4.1.13 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 74)
- A project is open  
See Opening a project (Page 97)
- PLC is not online

## Application

Using Openness API, you can use a new import mode for STEP7 objects to import blocks and UDTs even if a related object is missing.

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

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 SWImportOptions. 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
}
... // All kinds of blocks
PlcBlockComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreMissingReferencedObject);
...
... // UDTs
PlcTypeComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreMissingReferencedObject);
...
```

## See also

[Connecting to the TIA Portal \(Page 74\)](#)

[Opening a project \(Page 97\)](#)



### 8.4.1.14 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 74)
- A project is open  
See Opening a project (Page 97)
- 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
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, you can use SWImportOptions.IgnoreStructuralChanges, even if there are structural change and data loss.

```
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 74)
- Opening a project (Page 97)

## 8.4.2 Tag tables

### 8.4.2.1 Exporting PLC tag tables

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### 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 415)

### 8.4.2.2 Importing PLC tag table

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

## 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 41)

### 8.4.2.3 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 74)
- A project is open.  
See Opening a project (Page 97)

#### Application

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.

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) return;

    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) return;

    plcConstant.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml",
plcConstant.Name)), ExportOptions.WithDefaults);
}
```

## See also

Exporting configuration data (Page 415)

Notes on performance of TIA Portal Openness (Page 41)

### 8.4.2.4 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 74)
- A project is open.  
See Opening a project (Page 97)

#### Application

You can import either tags or constants in an import call.

---

**Note**

Constants can only be imported 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 415)

Notes on performance of TIA Portal Openness (Page 41)

**8.4.3 Exporting user data type**

**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)
- 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);
}
```

### 8.4.4 Importing user data type

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
Connecting to the TIA Portal (Page 74)
- A project is open.  
Opening a project (Page 97)
- PLC is not online.

#### Application

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="myudt1">
    <Sections>
      <Section Name="None">
        <Member Name="MyUDT1Member1" Datatype="bool"/>
        <Member Name="MyUDT1Member2" Datatype="myudt1">
          <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 ImportUserDataTypes(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 417)

**8.4.5 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 74)
- A project is open.  
See Opening a project (Page 97)
- PLC is not online

**Application**

You can export PLC data as OPC UA XML file by invoking an action on the TIA Portal Openness API. 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.

```
//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)));
}
```

## 8.5 Importing/exporting hardware data

### 8.5.1 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, October 2014
- Whitepaper AutomationML Part 2 –AutomationML Role Libraries, October 2014
- Whitepaper AutomationML Part 4 –AutomationML Logic, May 2010
- Whitepaper AutomationML– AutomationML Communication, September 2014
- Whitepaper AutomationML– AutomationML and eCI@ss Integration, November 2015
- Application Recommendations: Automation Project Configuration - AR\_001E Version 1.0.0, Mai.2017

#### Schema

The AutomationML data exchange model is described by the CAEX schema Version 2.15. You can download this file from the homepage of AutomationML e.V. (<https://www.automationml.org/o.red.c/dateien.html>)

### 8.5.2 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.

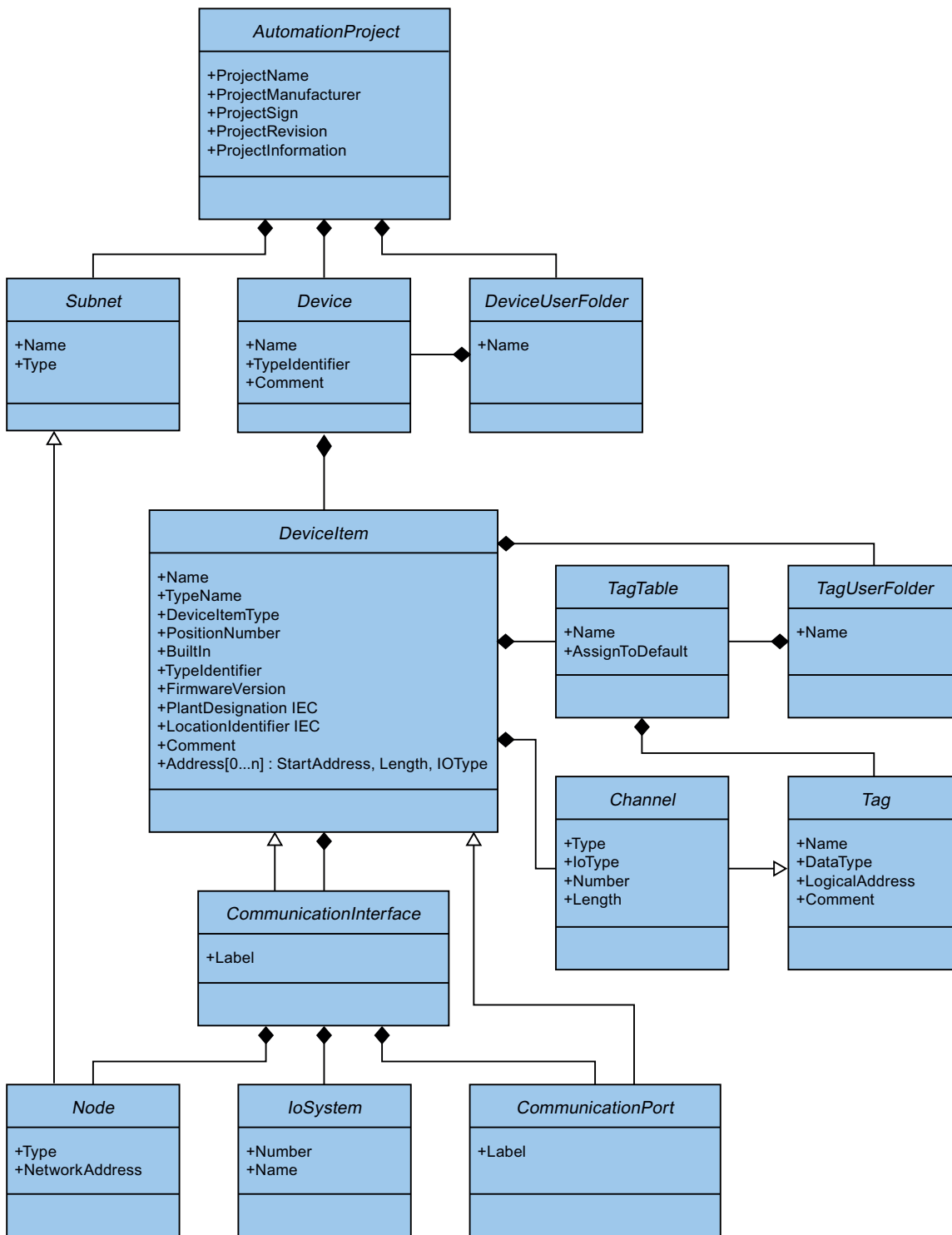
1. If a device item is pluggable, it shall not be pruned.
2. If a device item is of type "interface" or "port", it shall not be pruned.

3. AddressObjects of type "diagnosis" are not relevant for the prune algorithm.
4. 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).
5. Address objects shall be included in the same sequence as returned by TIA Portal Openness.

### **8.5.3 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.



## **8.5.4 Structure of the CAx data for importing/exporting**

### **Basic structure of an export file**

The data in the export file from the import/export is structured with reference to a basic structure. The export file is generated in an AML format.

The AML file starts with a document information. The file includes the data of the computer-specific installation of the exported project.

The export file comprises the following two sections:

- Additional information

The <WriterHeader> includes information about the export or import process. The import ignores the content of the <AdditionalInformation> section.

For example you can insert a <AdditionalInformation>...</AdditionalInformation> block, in which you place additional information about the validation. After the AML file is forwarded, the recipient can use this block before the import to check whether the AML file has been changed.

```
<xml version="1.0" encoding="utf-8">
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  FileName="CAx_asterisk_AML_03_V14.aml" SchemaVersion="2.15"
  xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
  <AdditionalInformation>
    <WriterHeader>
      <WriterName>Totally Integrated Automation Portal</WriterName>
      <WriterID>1d4fceb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
      <WriterVendor>Siemens AG</WriterVendor>
      <WriterVendorURL>www.siemens.com</WriterVendorURL>
      <WriterVersion>1400</WriterVersion>
      <WriterRelease>1400.100.101.16</WriterRelease>
      <LastWritingDateTime>2016-09-29T11:21:34.9551066Z</LastWritingDateTime>
    </WriterHeader>
  </AdditionalInformation>
  <AdditionalInformation AutomationMLVersion="2.0" />
  <AdditionalInformation DocumentVersions="Recommendations">
    <Document DocumentIdentifier="AR APC" Version="1.0.0" />
  </AdditionalInformation>
  ...
</CAEXFile>
```

- 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 individually 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 555)

## 8.5.5 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 physical 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

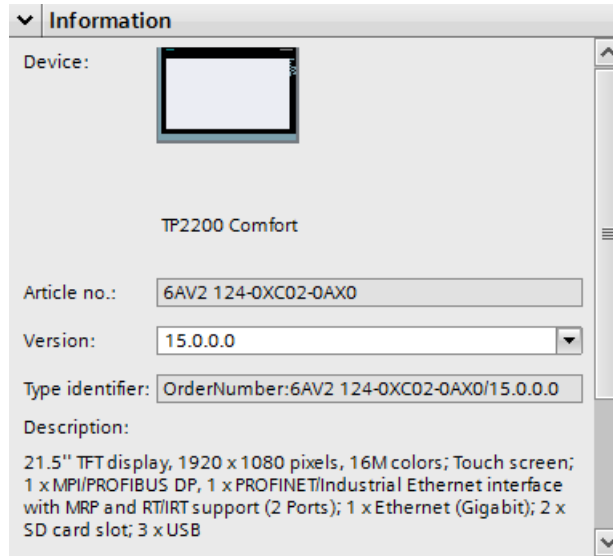
`System.` is the identifier for objects that cannot be determined by any other identifier. The formats for this `TypeIdentifierType` are as following:

- <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"



### 8.5.6 Export of CAx data

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

#### Application

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.

TIA Portal Openness provides the following ways to export CAx data:

- Export function  
The export function is accessed via `CaxProvider` service. To get the `CaxProvider` service invoke the `GetService` method at `Project` object.
- Command line interface  
You execute the "Siemens.Automation.Cax.AmiHost.exe" located in "C:\Program Files\Siemens\Automation\Portal V..\Bin\" by passing specific command line arguments.

### Export and Import restrictions for CAx

CAx does not support the export and import of

- Port-Port connections
- Connections to and between extension racks
- Multi-CPU's
- H-devices
- HMI devices except push button panels and key panels
- Drives
- Output mode and range of analog channels
- Packed addresses

CAx does not support the export and import of the following device and drives:

- 6AV2 104-0XXXX-XXXX
- 6AV2 155-0XXXX-XXXX
- 6ES7 XXX-XXXXX-XXXX
- 6ES7 370-0AA01-0AA0
- 6ES7 451-3AL00-0AE0
- 6GK5 414-3FC00-2AA2
- 6GK5 414-3FC10-2AA2
- 6GK5 495-8BA00-8AA2
- 6GK5 496-4MA00-8AA2
- 6GK5 602-0BA00-2AA3
- 6GK5 602-0BA10-2AA3
- 6GK5 612-0BA00-2AA3
- 6GK5 612-0BA10-2AA3
- 6GK5 613-0BA00-2AA3
- 6GK5 623-0BA10-2AA3
- 6GK5 627-2BA10-2AA3
- System:Device.Scalance/S627

- System:IPIProxy.Device
- System:IPIProxy.Rack

### Program code: Access the CaxProvider service

Modify the following program code to access the CaxProvider service:

```
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, use the `Export` method with the following parameters:

Name	Example	Description
ProjectToExport	tiaPortal.Projects[0]	Project object to Export
ExportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	Full Export file path of AML file
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	Full file path of Log file

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 at device level

To export CAx data at device level, use the `Export` method with the following parameters:

Name	Example	Description
DeviceToExport	project.Devices[0]	Device object to Export
ExportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	Full Export file path of AML file
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	Full file path of Log file



Modify the following program code to export CAx data at project level:

```
caxProvider.Export(device, new FileInfo(@"D:\Temp\DeviceExport.aml"), new
FileInfo(@"D:\Temp\DeviceExport_Log.log"));
```

### CAx export via command line

To export CAx data via command line, use "Siemens.Automation.Cax.AmiHost.exe" with the following parameters:

Parameter	Example	Description
-p	-p "D:\Temp\MyProject.ap14"	Specifies a full path name to an existing TIA Portal project.
-d	-d "S7300/ET200M station_1"	Optional paramter. If no device is speicified export will take place at project level. Specifies the name of the device or station inside the specified TIA project, that needs to be exported.
-m	-m "AML"	Specifies the export/import mode (format for export/import): "AML" exports in AML format
-e	-e "D:\Import" -e "D:\Import\CAx_Export.aml"	Specifies full path of AML file to be exported. The project name will be used as exported file name if only a path is specified.

Modify the following program code to o export CAx data at project level via the command line:

```
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.ap14" -m "AML" -e
"D:\Import\CAx_Export.aml"
```

Modify the following program code to o export CAx data at device level via the command line:

```
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.ap14" -d "S7300/
ET200M station_1" -m "AML" -e "D:\Import\CAx_Export.aml"
```

## 8.5.7 Export/Import of sub modules

### Requirements

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open  
See Opening a project (Page 97)
- 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 AML file structure that shall be generated during the export of sub modules.

```
<?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>1d4fceb6-1ad6-4881-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="1.0.0" />
    </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>
      </InternalElement>
    </InstanceHierarchy>
    ...
```

## Import sub modules

You can import sub modules from an AML files, which is generated from above export.

### Note

- The hierarchy in AML file shall not influence/affect the TIA Portal internal hierarchy after Import.
- The AML files created using older TIAP 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 TIAP hierarchy to required AML file hierarchy. In this example from TIAP 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:

- 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/loSystem, 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 loSystem 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.

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="MultipleBA_01.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
  <AdditionalInformation>
    <WriterHeader>
      <WriterName>Totally Integrated Automation Portal</WriterName>
      <WriterID>1d4fceb-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="1.0.0" />
  </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>
    </InternalElement>
  </InstanceHierarchy>
  ...

```

## 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 546)).

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 74)

Opening a project (Page 97)

## 8.5.8 Import of CAx data

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)

### 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.

TIA Portal Openness provides the following ways to export CAx data:

- Import function  
The import function is accessed via `CaxProvider` service. To get the `CaxProvider` service invoke the `GetService` method at `Project` object.
- Command line  
You execute the "Siemens.Automation.Cax.AmiHost.exe" located in "C:\Program Files\Siemens\Automation\Portal V..\Bin\" by passing specific command line arguments:

### 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 use the `Import` method with the following parameters:

Name	Example	Description
ImportFilePath	<code>new FileInfo(@"D:\Temp\ProjectExport.aml")</code>	Full import file path of AML file
LogFilePath	<code>new FileInfo(@"D:\Temp\ProjectExport_Log.log")</code>	Full file path of log file
ImportOptions	<code>CaxImportOptions.MoveToParkingLot</code> <code>CaxImportOptions.RetainTiaDevice</code> <code>CaxImportOptions.OverwriteTiaDevice</code>	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

## CAx import via command line

To import CAx data via command line, use "Siemens.Automation.Cax.AmiHost.exe" with the following parameters:

Parameter	Example	Description
-p	<code>-p "D:\Temp\MyProject.ap14"</code>	Specifies a full path name to an existing TIA Portal project.
-m	<code>-m "AML"</code>	Specifies the export/import mode (format for export/import): "AML" exports in AML format
-i	<code>-i "D:\Import\CAx_Export.aml"</code>	Specifies full path of the AML file to be imported.
-c	<code>-c "ParkingLot"</code>	Specifies different strategies if there is a conflict in the device name according to import options.

Modify the following program code to o import CAx data via the command line:

```
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.ap14" -m "AML" -i "D:\Import\CAx_Export.aml"
```

The following import options are provided:

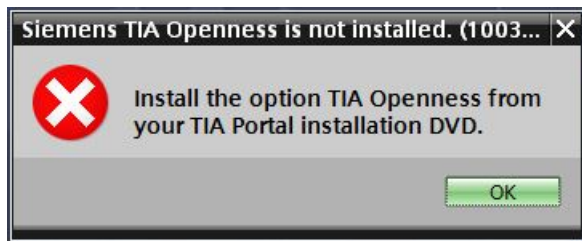
Import option	Description
ParkingLot	Retain name conflicting device/s in the project and import those out of CAx into a parkinglot folder
RetainTia	Retain name conflicting device/s in the project and do not import those out of CAx
OverwriteTia	Overwrite name conflicting device/s in the project by the ones out of CAx

### 8.5.9 Exceptions during import and export of CAx data

#### Exception due to non-availability of TIA Openness

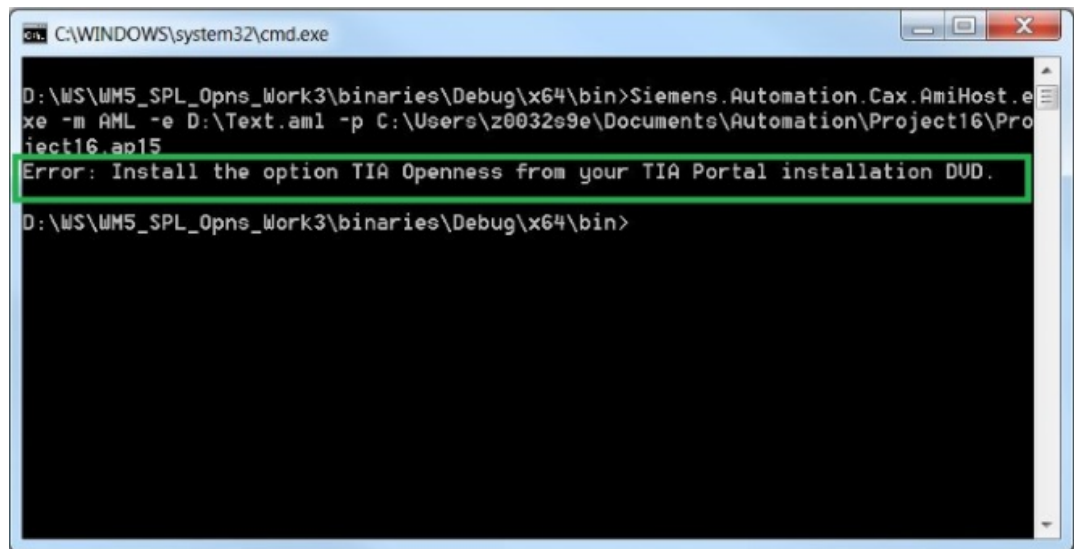
CAx implementation is based on TIA Openness Public API's. Openness Public API's are available only when user has installed Openness optional pack during TIA Portal installation. Hence, there is a need to check whether Openness is available before performing any CAx related functionalities. (Refere Installing TIA Openness (Page 27))

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.



While performing the CAx operation through command-line, the following error dialog displays during the non-availability of TIA Openness.



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows a command being executed: "D:\MS\WM5\_SPL\_Opns\_Work3\binaries\Debug\x64\bin>Siemens.Automation.Cax.AmiHost.exe -m AML -e D:\Text.aml -p C:\Users\z0032s9e\Documents\Automation\Project16\Project16.ap15". The output of the command is "Error: Install the option TIA Openness from your TIA Portal installation DVD.", which is highlighted with a green rectangular box. The prompt then returns to "D:\MS\WM5\_SPL\_Opns\_Work3\binaries\Debug\x64\bin>".

```
C:\WINDOWS\system32\cmd.exe
D:\MS\WM5_SPL_Opns_Work3\binaries\Debug\x64\bin>Siemens.Automation.Cax.AmiHost.exe -m AML -e D:\Text.aml -p C:\Users\z0032s9e\Documents\Automation\Project16\Project16.ap15
Error: Install the option TIA Openness from your TIA Portal installation DVD.
D:\MS\WM5_SPL_Opns_Work3\binaries\Debug\x64\bin>
```

Figure 8-3 OpennessNotInstalled-Commandline

## 8.5.10 Round trip exchange of devices and modules

### Requirement

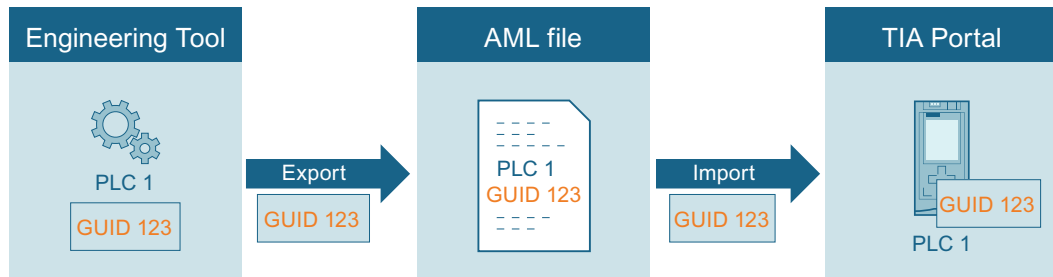
- The TIA Portal Openness application is connected to the TIA Portal. See Connecting to the TIA Portal (Page 74)
- A project is open. See Opening a project (Page 97)
- PLC is offline.

### 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, ...

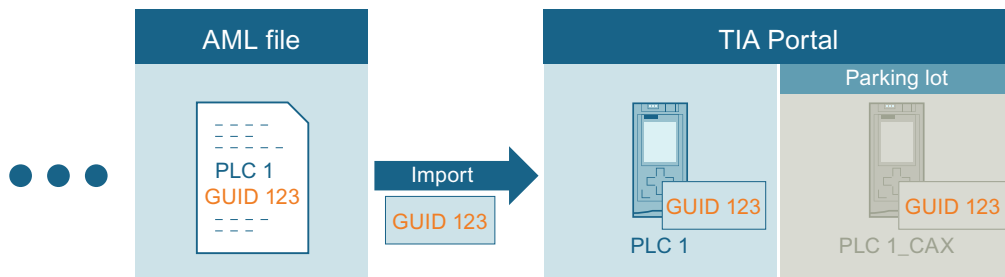
During the first export from the TIA Portal, the AML GUID for a device or a no built-in device item is randomly generated, but kept stable afterwards.



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 corresponding 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".



**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.

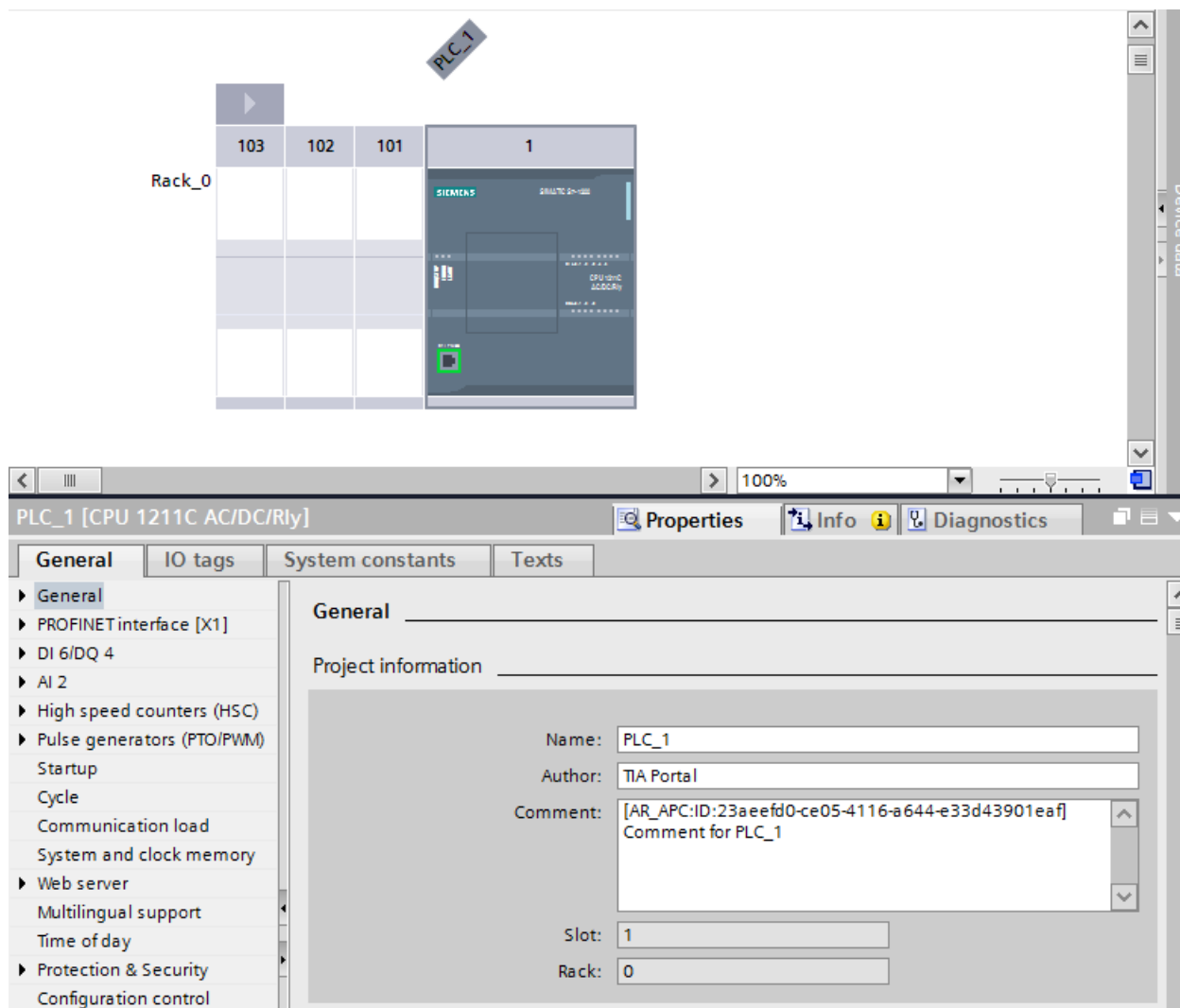
**Import settings**

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".

**Note****Valid AML GUID**

If you edit an AML GUID before the import, the AML GUID becomes invalid and the import will be aborted.

After the import into the TIA Portal, the AML GUID is added to the existing user comments as follows:



---

**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.

---

**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"
```

**8.5.11 Export/Import topology**

**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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.

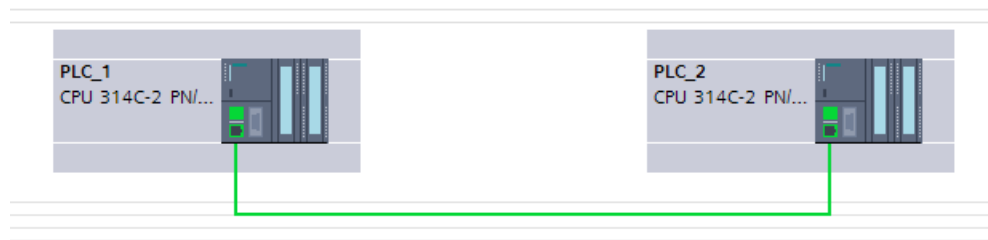
<InterLink> element gives link details of port to port interconnection between the device items. It appears under the common parent of the connected devices, and contains unique tag names.

## 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 <InternalLink> 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" />

```

### 8.5.12 Export of a device object

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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.

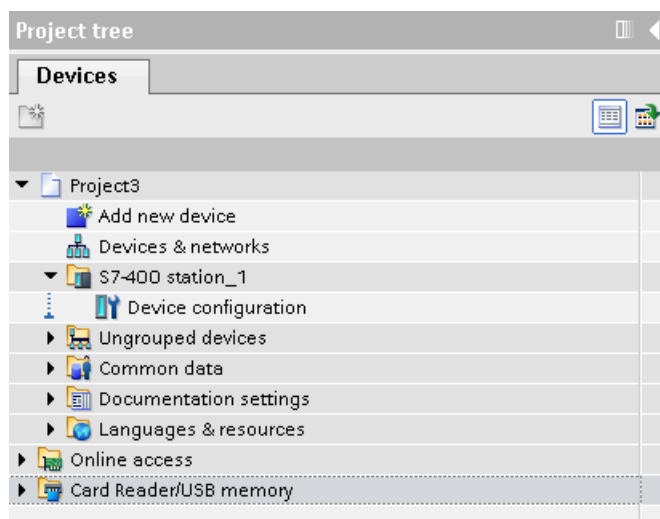
---

#### Attributes

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	
TypeIdentifier	Mandatory	
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>

```

## See also

Structure of the CAx data for importing/exporting (Page 550)

AML type identifiers (Page 555)

### 8.5.13 Import of a device object

#### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- 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 unassigned devices 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:

▼	Group_1	
▼	Group_2	
▼	Group_3	
▼	Group_4	

### See also

Structure of the CAx data for importing/exporting (Page 550)

AML type identifiers (Page 555)

## 8.5.14 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 74)
- A project is open.  
See Opening a project (Page 97)
- 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.

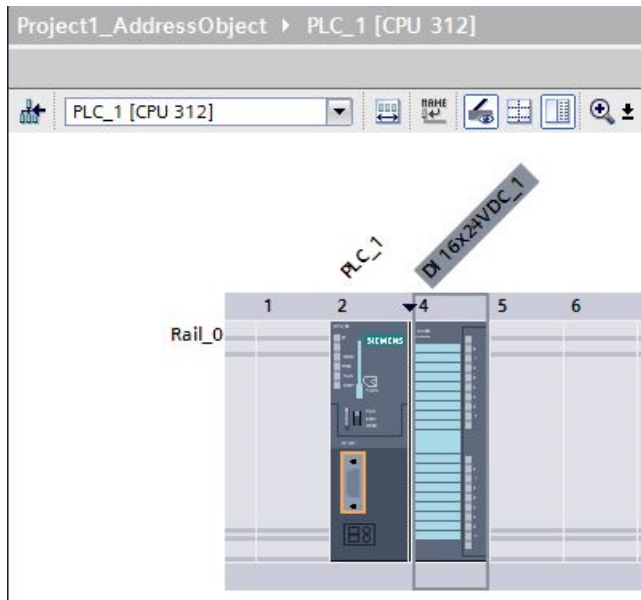
The Address attribute in the AML file contains RefSemantic mandatorily set to the specified value named OrderedListType.

### Attributes of a "Address" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Io-Type	Mandatory	Input or Output
Length	Optional	Channel width
StartAddress	Mandatory	Start address of the IO device.

**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 elements 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>
  </Attribute>
</InternalElement>
```

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 546)

### 8.5.15 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 74)
- A project is open.  
See Opening a project (Page 97)
- 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

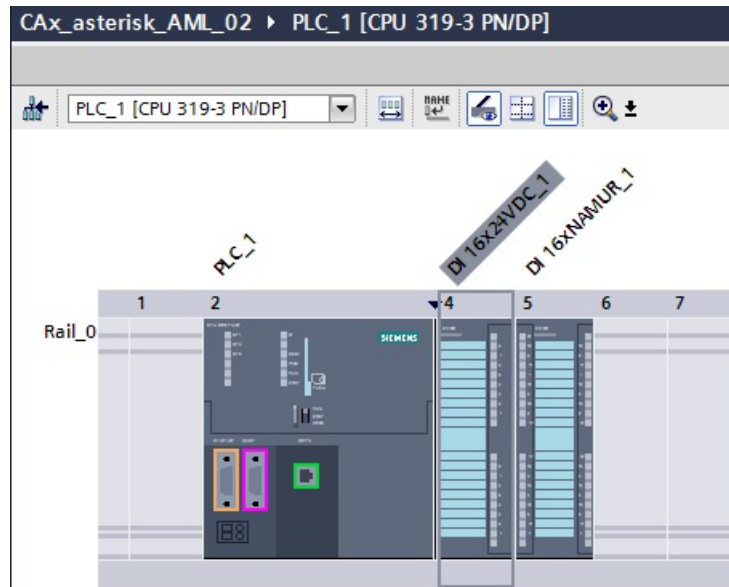
The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Io-Type	Mandatory	Input or Output
Length	Optional	Channel width (1 for Digital and 16 for Analog signals)
Number	Mandatory	Channel number starts from 0
Type	Mandatory	Analog or Digital

#### Channel numbering

Digital Input, Digital Output, Analog Input, Analog Output, 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 figure 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>
```

## 8.5.16 Export of device item objects

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See Connecting to the TIA Portal (Page 74)
- A project is open.  
See Opening a project (Page 97)
- The PLC is offline.

### Application

The export of device item objects 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).
PositionNumber	Mandatory	
BuiltIn	Optional	Default: FALSE
TypeIdentifier	Mandatory for "BuiltIn" = FALSE Ignored for "BuiltIn" = TRUE	
FirmwareVersion	Optional, mandatory if the object supports firmware versions	
PlantDesignation IEC	Optional	Default: ""

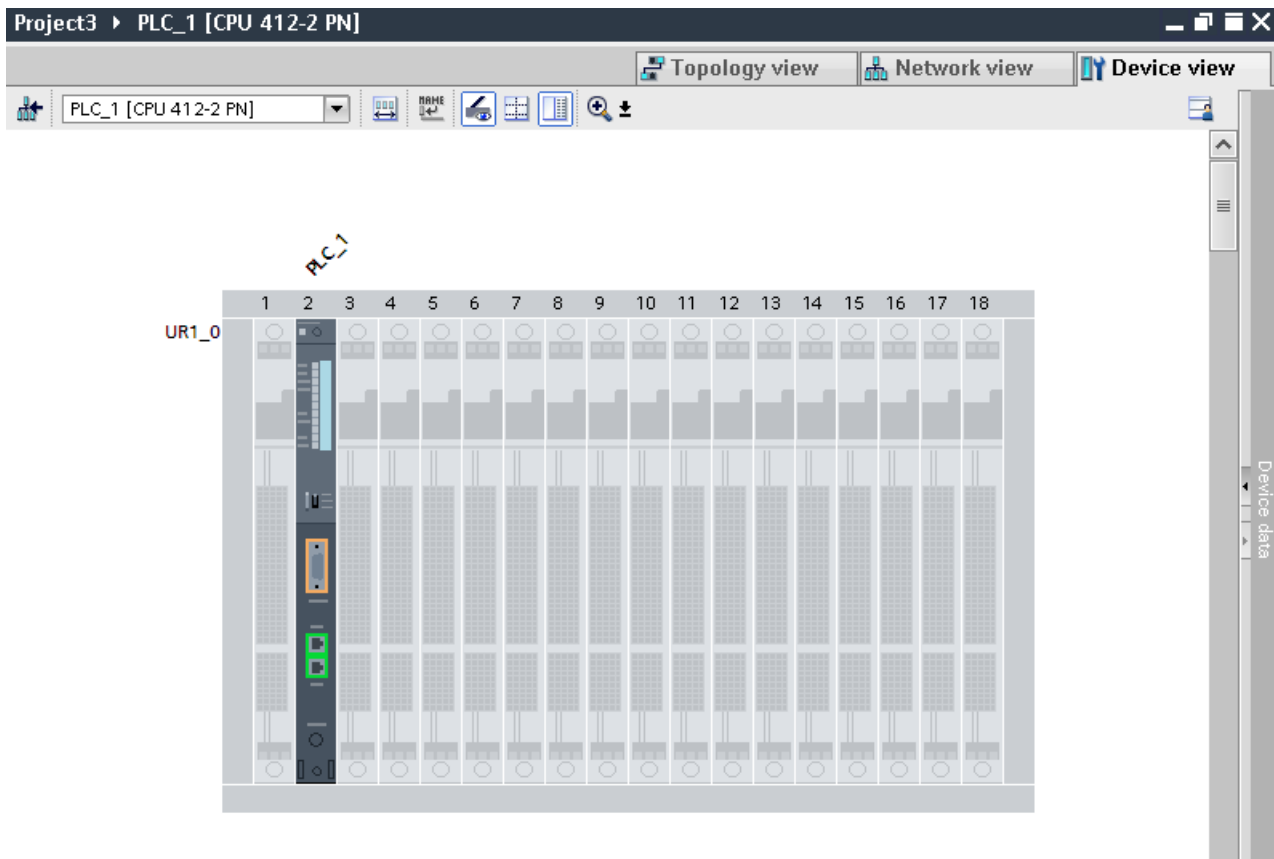


Attribute	Handling	Comment
LocationIdentifier IEC	Optional	Default: ""
Comment	Optional for "BuiltIn" = FALSE	Default: ""
Address	Optional	"Address" has nested attributes

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="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>OrderNumber:6ES7 400-1TA01-0AA0</Value>
  </Attribute>
<InternalElement ID="alde449e-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>
...

```

## See also

- Export/Import of GSD/GSDML based devices and device items (Page 590)
- Structure of the CAx data for importing/exporting (Page 550)
- AML type identifiers (Page 555)

## 8.5.17 Import of device item objects

### Requirement

- The TIA Portal Openness application is connected to the TIA Portal.  
See [Connecting to the TIA Portal \(Page 74\)](#)
- A project is open.  
See [Opening a project \(Page 97\)](#)
- The PLC is offline.

### Application

The import of device item objects 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 import 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".

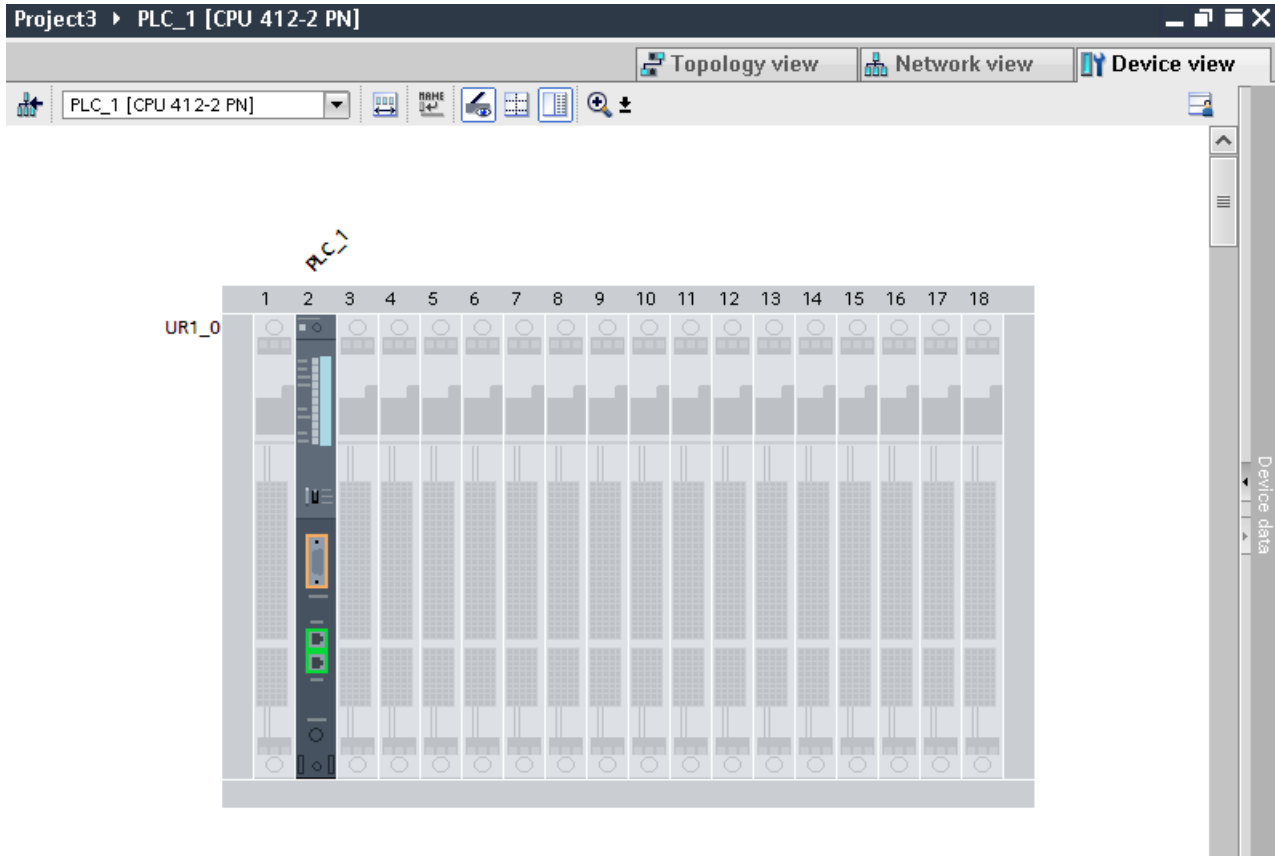
```

...
<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="alde449e-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 550)
- AML type identifiers (Page 555)

### 8.5.18 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
- A project is open.  
See Opening a project
- 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 import	
Comment	Optional for import	

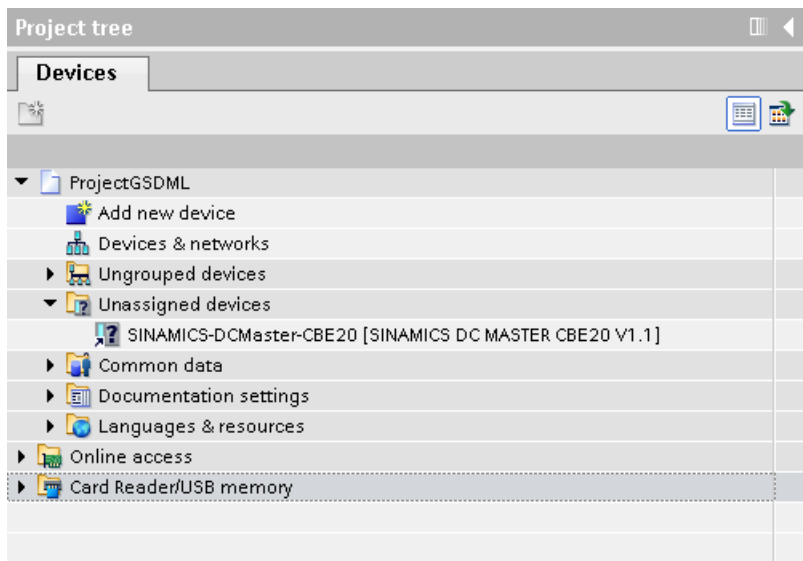
## Attributes for a device item

The following table shows the related attributes of a device item for CAx import and export files:

Attribute	Handling for attribute		Comment
	BuiltIn = FALSE Generic device items	BuiltIn = TRUE Physical device items	
Name	Mandatory	Export-only	
TypeName	Export-only	Not applicable	
DeviceItem-Type	Export-only	Export-only	Only for PLC (central devices) und Head-Module (decentral devices) device items

Attribute	Handling for attribute BuiltIn = FALSE Generic device items	Handling for attribute BuiltIn = TRUE Physical device items	Comment
PositionNumber	Mandatory	Mandatory for export Exceptional cases: Device item type interface: Optional for import Device item type port: Mandatory for import of builtIn devices if "Label" attribute is not specified. If both 'PositionNumber' and 'Label' are configured, then 'PositionNumber' gets higher precedence for export and import.	
BuiltIn	Optional		Default: FALSE
TypeIdentifier	Mandatory for "BuiltIn" = FALSE	Ignored for "BuiltIn" = TRUE	
Comment	Optional	Not applicable	
Label	-	- Device item type interface: Mandatory Device item type port: Mandatory if 'PositionNumber' attribute is not specified. If both 'PositionNumber' and 'Label' are configured, then 'PositionNumber' gets higher precedence and same shall be considered for import.	

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>
...

```

**See also**

AML type identifiers (Page 555)

**8.5.19 Export/Import of subnets**

**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
- 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

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	
FirmwareVersion	Mandatory	
TypeName	Export-only	No relevance for "BuiltIn" device items.
DeviceItem Type	Export-only	Only for for CPU and HeadModule
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	Label may be missing if "BuiltIn" = TRUE and "PositionNumber" are specified for the related "DeviceItem" object.
TypeIdentifier	Mandatory	
FirmwareVersion	Mandatory	
TypeName	Export-only	No relevance for "BuiltIn" device items.
DeviceItem Type	Export-only	Only for for CPU and HeadModule.
PositionNumber	Mandatory	Only relevant for the import of "BuiltIn" device items, if "Label" attribute is not specified. If both "PositionNumber" and "Label" are configured, then "PositionNumber" gets higher precedence.
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:

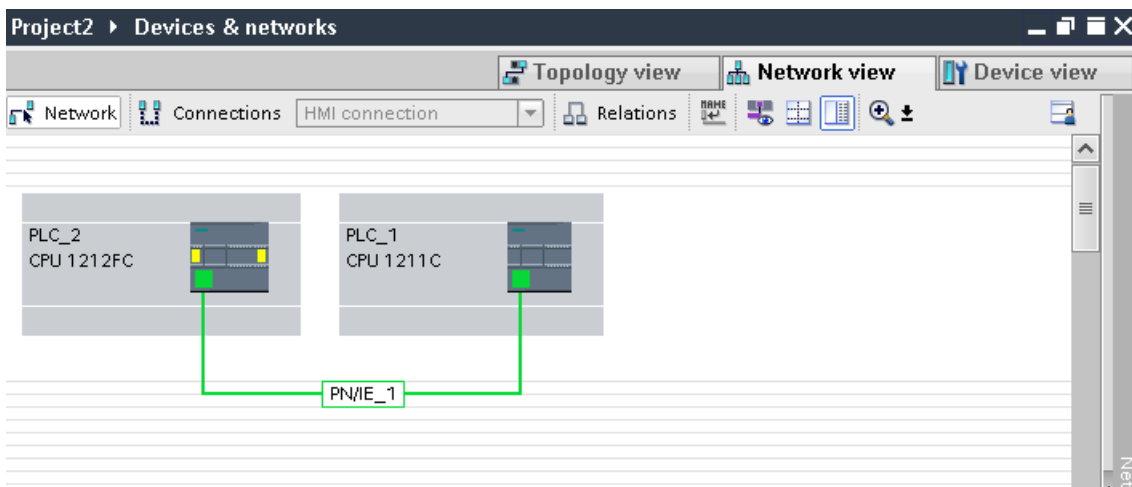
Attribute	Handling	Comment
Name	Export-only	MPI, PROFIBUS, PROFINET
Type	Export-only	Ethernet or PROFIBUS or MPI or ASi
NetworkAddress	Mandatory	
SubnetMask	Optional	PROFINET For import, default value is retained if no value is set.
RouterAddress	Optional	PROFINET For import, default value is retained 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

## 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-a703-1acf73e5f3" 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>
    </InternalElement>
  </InternalElement>
</InstanceHierarchy>
...

```

```

...
<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</Value>
  </Attribute>
  <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-b846-30dad8726d1"
      Name="CommunicationPortInterface"
      RefBaseClassPath=
        "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
...

```

## See also

- Structure of the CAx data for importing/exporting (Page 550)
- Connecting to the TIA Portal (Page 74)
- Opening a project (Page 97)

## 8.5.20 Export/Import of PLC tags

### 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
- 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>`  
Tag 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	Mandatory	Imported and exported in international mnemonics format
Comment	Optional	

### 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="a310ba93-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="f9269ce4-c015-459f-9f59-8f94bca3b186" Name="Tag table_1">
    <ExternalInterface ID="fc0c8c5a-fd5b-443b-b430-6435b6aa22ff" Name="Tag_1"
      RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
      ....
    </ExternalInterface>
    <ExternalInterface ID="450d6a1d-81b8-49ae-a104-c0072933d669" Name="Tag_2"
      RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
      ....
    </ExternalInterface>
    <ExternalInterface ID="3de17a36-b5c5-4fc7-9fc3-47e4a8f95087" Name="Tag_3"
      RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
      <Attribute Name="DataType" AttributeDataType="xs:string">
        <Value>Word</Value>
      </Attribute>
      <Attribute Name="LogicalAddress" AttributeDataType="xs:string">
        <Value>IWO</Value>
      </Attribute>
    </ExternalInterface>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/TagTable" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
</InternalElement>
...

```

**See also**

- Structure of the CAx data for importing/exporting (Page 550)
- Connecting to the TIA Portal (Page 74)
- Opening a project (Page 97)

**8.5.21 Export/Import of IO-systems****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
- 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.

### 8.5.22 Export/Import of multilingual comments

#### 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
- 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.

English (United States)	French (France)	German (Germany)	Chinese (People's Republic of Chi...	Reference
Profinet_Module	Profinet_Module_fr	Profinet_Module_de	Profinet_Module_cs	PROFINET interf...
Device_01	machine_01	Gerät_01	Device_01_chs	S7-1200 statio...
				Project2\PLC_1...
				Project2\PLC_1...

### 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 550)
- Connecting to the TIA Portal (Page 74)
- Opening a project (Page 97)

## 8.5.23 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 8-6 Attribute names of devices and GSD/GSDML devices

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
Comment	Comment

Table 8-7 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)	<b>Classification</b>
PositionNumber	PositionNumber
BuiltIn	<b>IsBuiltIn</b>
PlantDesignation IEC	<b>PlantDesignation</b>
LocationIdentifier IEC	<b>LocationIdentifier</b>
Comment	Comment

Table 8-8 Attribute names of GSD/GSDML device items

<b>AML file</b>	<b>TIA Portal Openness</b>
Name	Name
TypeIdentifier	TypeIdentifier
TypeName	TypeName
DeviceItem Type (for HeadModule)	<b>Classification</b>
PositionNumber	PositionNumber
BuiltIn	<b>IsBuiltIn</b>
Comment	Comment
Label	Label

Table 8-9 Attribute names of tags

<b>AML file</b>	<b>TIA Portal Openness</b>
Name	Name
DataType	<b>DataTypeName</b>
LogicalAddress	LogicalAddress
Comment	Comment

Table 8-10 Attribute names of tag tables

<b>AML file</b>	<b>TIA Portal Openness</b>
Name	Name
AssignToDefault	<b>IsDefault</b>

Table 8-11 Attribute names of addresses

<b>AML file</b>	<b>TIA Portal Openness</b>
StartAddress	StartAddress
Length	Length
IoType	IoType

Table 8-12 Attribute names of ports

<b>AML file</b>	<b>TIA Portal Openness</b>
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
PositionNumber	PositionNumber
BuiltIn	<b>IsBuiltIn</b>

8.5 Importing/exporting hardware data

AML file	TIA Portal Openness
Comment	Comment
Label	Label

Table 8-13 Attribute names of devices with IO-interface

AML file	TIA Portal Openness
Name	Name
TypIdentifier	TypIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
DeviceItem Type (for CPU and HeadModule)	<b>Classification</b>
PositionNumber	PositionNumber
BuiltIn	<b>IsBuiltIn</b>
Label	Label
Comment	Comment

Table 8-14 Attribute names of channels

AML file	TIA Portal Openness
Type	Type
IoType	IoType
Number	Not mapped to any attribute in TIA Portal Openness.
Length	<b>ChannelWidth</b>



# Major Changes

## 9.1 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 DeviceItemComposition 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.

## 9.2 Major changes in V14 SP1

### 9.2.1 Major changes in V14 SP1

#### Introduction

The following changes were made in TIA Portal Openness API object model V14 SP1, which may impact your existing applications:

Change	Required program code adjustment
Improved handling for master copies	<p>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.</p> <p>The following composition support CreateFrom:</p> <ul style="list-style-type: none"> <li>• Siemens.Engineering.HW.DeviceComposition</li> <li>• Siemens.Engineering.HW.DeviceItemComposition</li> <li>• Siemens.Engineering.SW.Blocks.PlcBlockComposition</li> <li>• Siemens.Engineering.SW.Tags.PlcTagTableComposition</li> <li>• Siemens.Engineering.SW.Tags.PlcTagComposition</li> <li>• Siemens.Engineering.SW.Types.PlcTypeComposition</li> <li>• Siemens.Engineering.SW.TechnologicalObjects.TechnologicalInstanceDBCComposition</li> <li>• Siemens.Engineering.SW.Tags.PlcUserConstantComposition</li> <li>• Siemens.Engineering.Hmi.Tag.TagTableComposition</li> <li>• Siemens.Engineering.Hmi.Tag.TagComposition</li> <li>• Siemens.Engineering.Hmi.Screen.ScreenComposition</li> <li>• Siemens.Engineering.Hmi.Screen.ScreenTemplateComposition</li> <li>• Siemens.Engineering.Hmi.RuntimeScripting.VBScriptComposition</li> <li>• Siemens.Engineering.HW.SubnetComposition</li> <li>• Siemens.Engineering.HW.DeviceUserGroupComposition</li> <li>• Siemens.Engineering.SW.Blocks.PlcBlockUserGroupComposition</li> <li>• Siemens.Engineering.SW.ExternalSources.PlcExternalSourceUserGroupComposition</li> <li>• Siemens.Engineering.SW.Tags.PlcTagTableUserGroupComposition</li> <li>• Siemens.Engineering.SW.Types.PlcTypeUserGroupComposition</li> </ul>
Improved handling for global libraries	<p>Existing actions on global libraries can now be modifying actions, e.g. delete a master copy from a global library.</p> <p>UpdateProject and UpdateLibrary do not longer use the UpdatePathsMode and DeleteUnusedVersionsMode parameters. Unused versions are not deleted after an update</p>

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: <ul style="list-style-type: none"> <li>• Open project</li> <li>• Open library</li> <li>• Create project</li> <li>• Create global library</li> <li>• ...</li> </ul>

### 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.

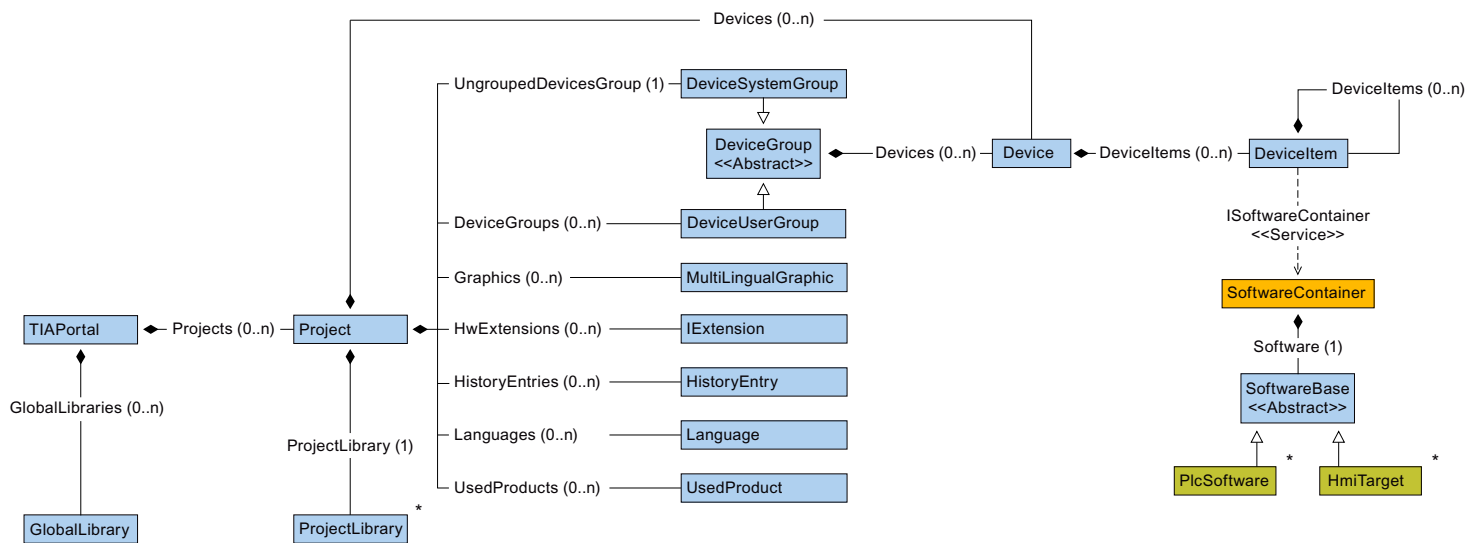
## 9.2.2 Major changes in the object model

### Object model of TIA Portal Openness V14

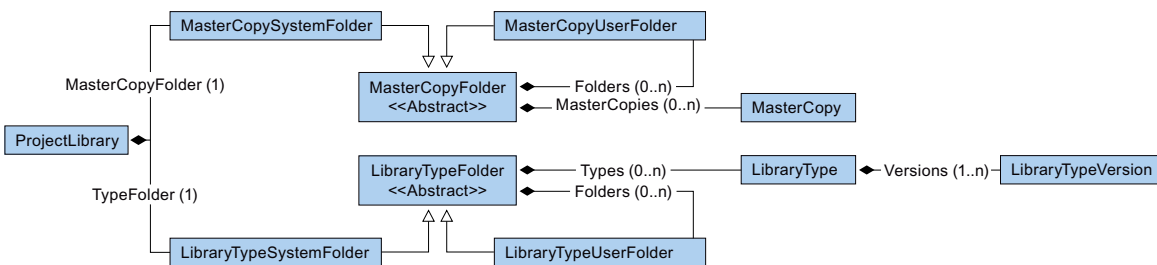
In order to allow you a comparison between the old and the new object model of TIA Portal Openness, the diagram below describes the object model of TIA Portal V14.

**Note**

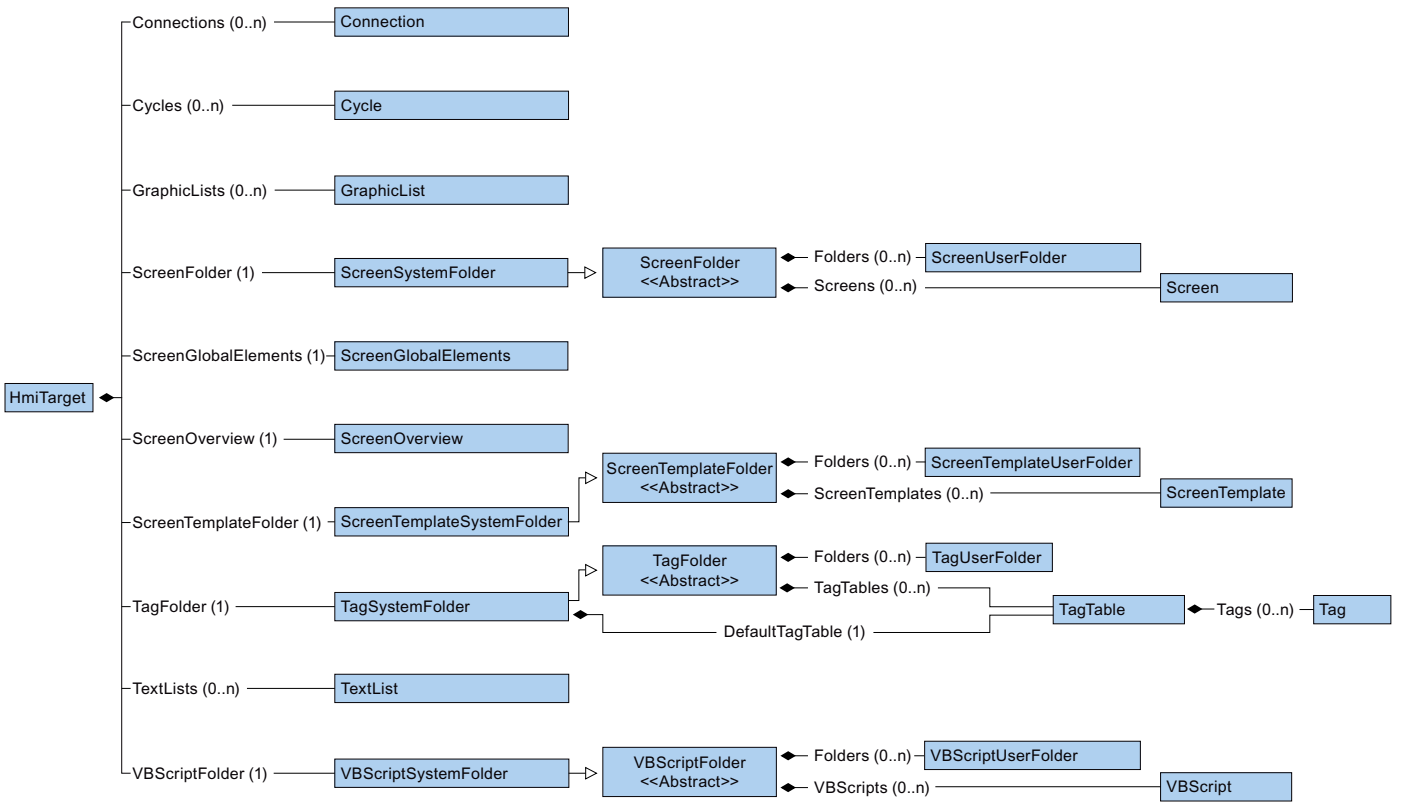
The object model described on the diagram is obsolete, for information about the object model of TIA Portal Openness V14 SP1 refer to TIA Portal Openness object model (Page 51)



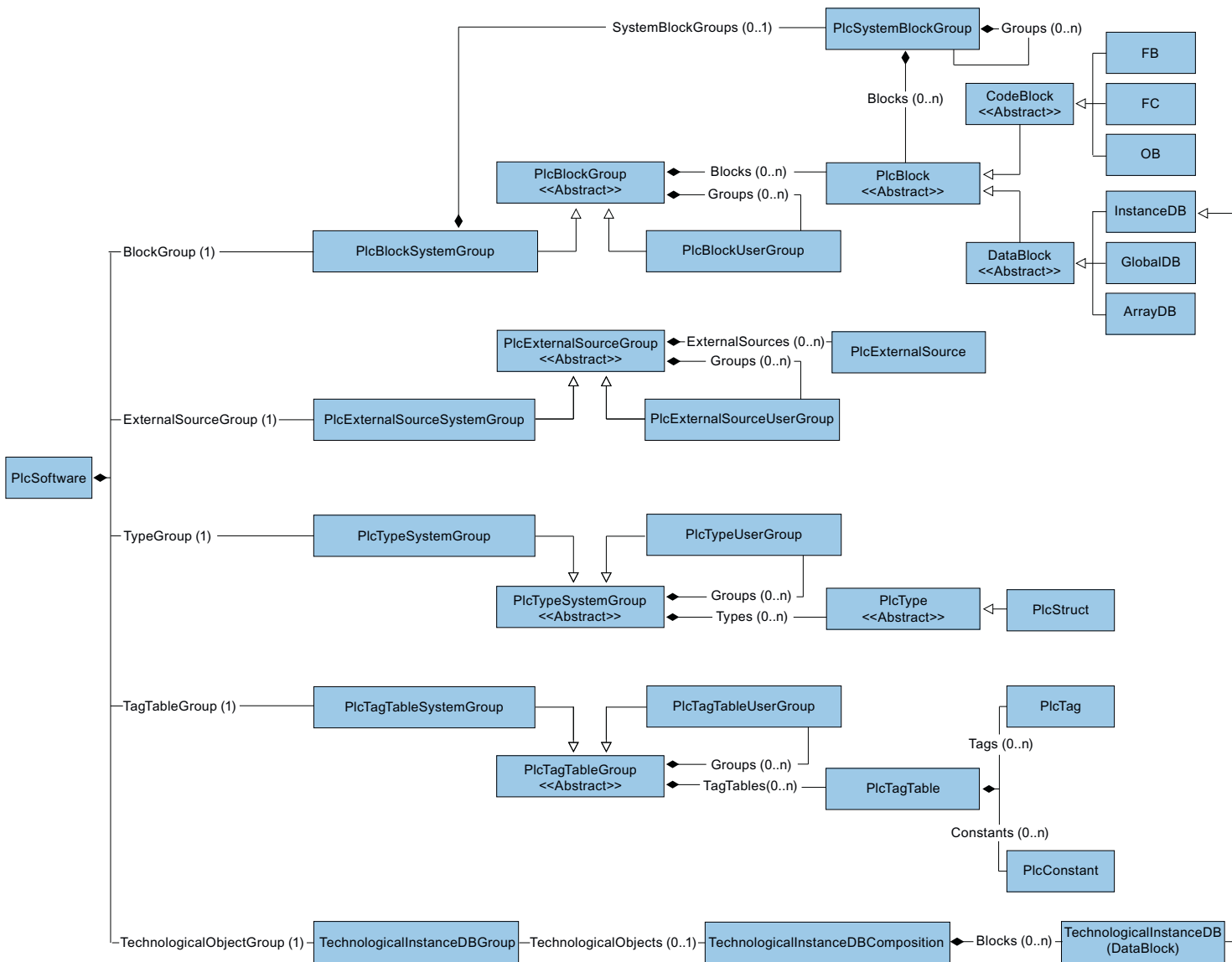
The following diagram describes the objects which are located under ProjectLibrary.



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



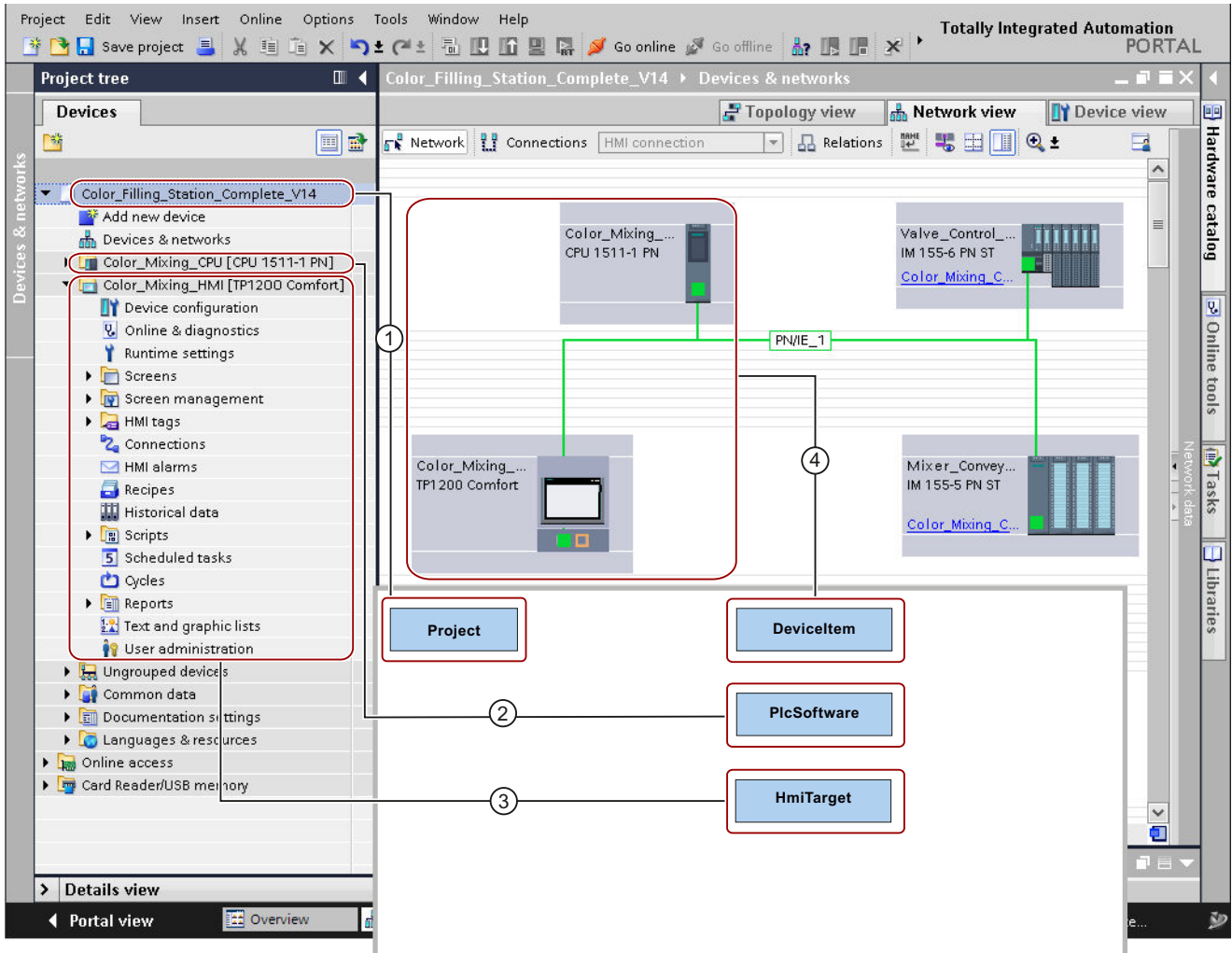
The following diagram describes the objects which are located under PlcSoftware.





## 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.

### 9.2.3 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.ModuleInformation-Provider	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.Attachment-Type		Siemens.Engineering.HW.MediumAttachment-Type	
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.ChannelloType	byte	Siemens.Engineering.HW.ChannelloType	int
Siemens.Engineering.HW.Enums.ChannelType	byte	Siemens.Engineering.HW.ChannelType	int
Siemens.Engineering.HW.Enums.DeviceItem-Classifications		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	ProfinetUpdateTime-Mode	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	
PnDeviceNameAutoGenerated		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	
OImP12		PbOImP12	
OImG12		PbOImP12	
OImG12Eec		PbOImG12Eec	
OImG121300		PbOImG121300	
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	

Attribut	Data type	new name	Data type
Tslot		PbTslot	
MinTsdr		PbMinTsdr	
MaxTsdr		PbMaxTsdr	
Tid1		PbTid1	
Tid2		PbTid2	
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		PbAdditionalPassivDeviceForIsochronousMode	
TotalDevice		PbTotalPassivDeviceForIsochronousMode	
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	

## Major Changes

### 9.2 Major changes in V14 SP1

Attribut	Data type	new name	Data type
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.ChannelloType

Attribut	Data type	new name	Data type
ChannelloType.Unknown		ChannelloType.Complex	

#### Modifications for attributes values of Siemens.Engineering.HW.IpProtocolSelection

Attribut	Data type	new name	Data type
IpProtocolSelection.AddressTailoring		IpProtocolSelection.VialoController	

#### 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	

Attribut	Data type	new name	Data type
TransmissionRateAndDuplex.X1000Mbps_FullDuplex		TransmissionRateAndDuplex.X1000MbpsFullDuplex	
TransmissionRateAndDuplex.FO1000Mbps_FullDuplex_LD		TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD	
TransmissionRateAndDuplex.FO1000Mbps_FullDuplex		TransmissionRateAndDuplex.FO1000MbpsFullDuplex	
TransmissionRateAndDuplex.TP1000Mbps_FullDuplex		TransmissionRateAndDuplex.TP1000MbpsFullDuplex	
TransmissionRateAndDuplex.FO10000Mbps_FullDuplex		TransmissionRateAndDuplex.FO10000MbpsFullDuplex	
TransmissionRateAndDuplex.FO100Mbps_FullDuplex_LD		TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	
TransmissionRateAndDuplex.POFPCF100Mbps_FullDuplex_LD		TransmissionRateAndDuplex.POFPCF100MbpsFullDuplexLD	

## 9.2.4 Changes for export and import

### 9.2.4.1 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

### 9.2.4.2 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

```
// 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");

PlcBlocksSystemGroup blocksGroup = ...;
blocksGroup.GenerateSourceFromBlocks(blocks, fileInfo);
PlcTypesSystemGroup 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);
```

### 9.2.4.3 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>

```

### 9.2.4.4 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	unordered	forbidden	mandatory	%I0.0
Q				%Q0.0
M				%M0.0

Area as of V14 SP1	Type	Block number	Bit offset	Example
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.LADFBFD.xsd**

CallRef node is renamed to Call and omits the BooleanAttribute subnode.

**InstructionRef node in SW.PlcBlocks.LADFBFD.xsd**

InstructionRef node is replaced by Part node

**Part node in SW.PlcBlocks.LADFBFD.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.LADFBFD.xsd**

Name attribe of Wire node removed.

### TemplateReference node in SW.PlcBlocks.LADFBFD.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

## 9.2.4.5 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 flagged 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

<pre> CALL  "Block_2"   Input_1  := "Tag_1"   Input_2  := "Tag_2"   Output_1 := "Tag_3"   Output_2 := "Tag_4" </pre>
--

### 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>
    </CallInfo>
  </Access>
</StlStatement>
...

```

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 constants

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 signed 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/FBD 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 insensitivity 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 exist.

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>
      </Sections>
    </Interface>
    ....
  <StlStatement UId="22">
    <StlToken Text="CALL" />
    <Access Scope="Call">
      <CallInfo BlockType="FB">
        <!-- Multiinstance 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>

```

### 9.2.4.6 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 InvalidValuesAcquisition 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

## **9.3 Major changes in V14**

### **9.3.1 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.

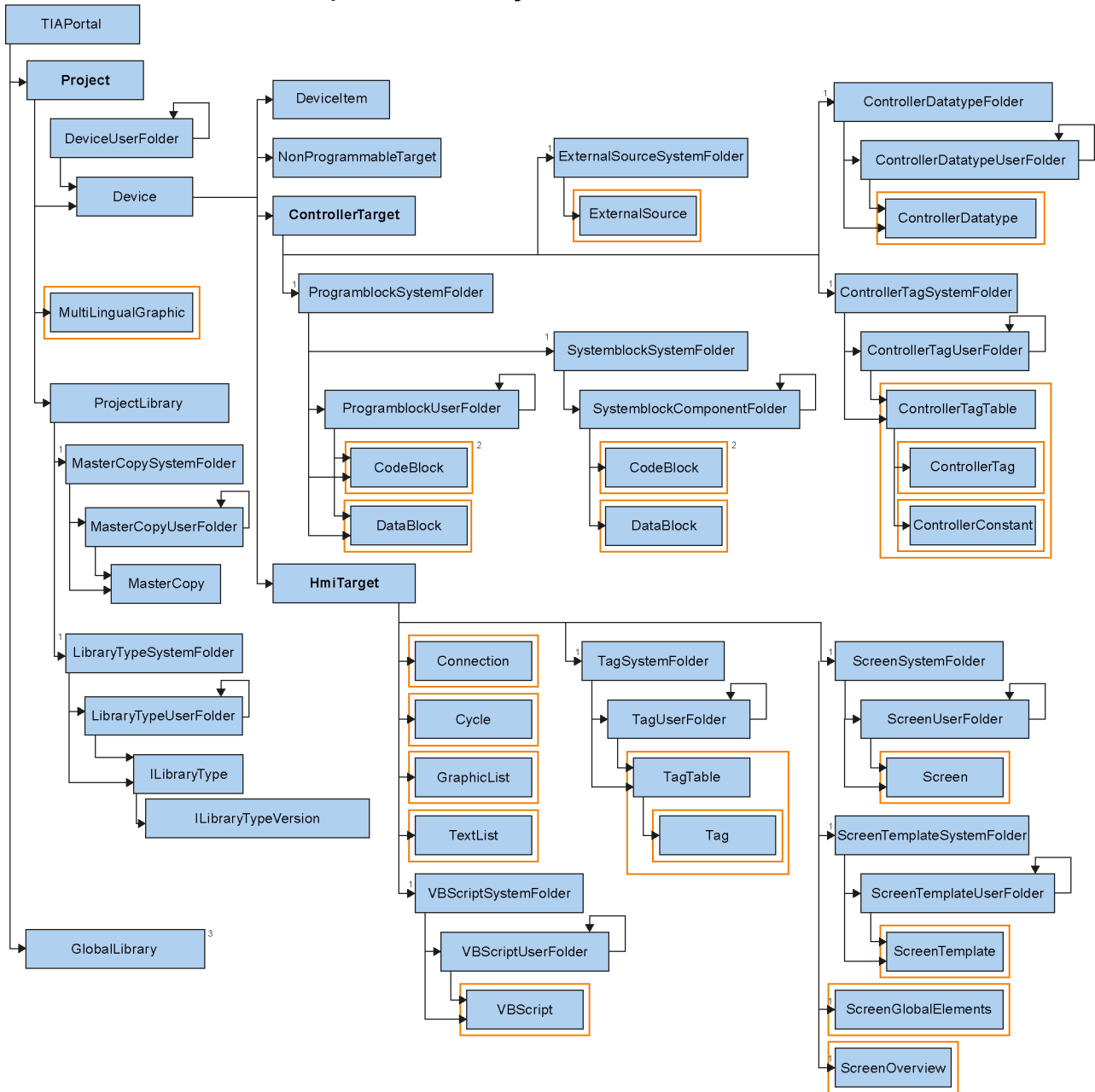
---

#### **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 51)

---

## Openness object model V13 SP1



## 9.3.2 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 the references to the V14 API by adding the following TIA Portal Openness APIs:
  - `Siemens.Engineering`
  - `Siemens.Engineering.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.

## 9.3.3 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: <ul style="list-style-type: none"> <li>• TIA Portal Openness V13 SP1(obsolete):  <pre>controllerTarget.Compile(CompilerOptions. Software, BuildOptions.Rebuild);</pre> </li> <li>• TIA Portal Openness V14:  <pre>plcSoftware.GetService&lt;ICompilable&gt;().Com pile();</pre> </li> </ul>
New namespaces have been added.	<ol style="list-style-type: none"> <li>1. Add the following namespace statements:  <pre>Siemens.Engineering.SW.Blocks; Siemens.Engineering.SW.ExternalSources; Siemens.Engineering.SW.Tags; Siemens.Engineering.SW.Types;</pre> </li> <li>2. Remove the <code>using ControllerTarget = Siemens.Engineering.HW.ControllerTarget</code> namespace statement.</li> <li>3. Compile the application.</li> </ol>

Change	Required program code adjustment
<p>ControllerTarget has been replaced by PlcSoftware and functionality has been changed in some cases.</p>	<ol style="list-style-type: none"> <li>1. Review the code examples in the documentation which belong to your application functionality.</li> <li>2. Update the program code of your TIA Portal Openness application according to the following example: <ul style="list-style-type: none"> <li>- TIA Portal Openness V13 SP1(obsolete): <pre>ControllerTarget controllerTarget = deviceItem as ControllerTarget</pre> </li> <li>- TIA Portal Openness V14: <pre>PlcSoftware plcSoftware = deviceItem.GetService&lt;SoftwareContainer &gt;().Software as PlcSoftware</pre> </li> </ul> </li> <li>3. Compile the application.</li> </ol>
<p>Objects have been replaced.</p>	<ol style="list-style-type: none"> <li>1. Search and replace the following objects: <pre>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 IOOnline = OnlineProvider ILibraryType = LibraryType</pre> </li> <li>2. Compile the application.</li> </ol>

Change	Required program code adjustment
Aggregations have been replaced by compositions.	<ol style="list-style-type: none"> <li>1. Replace every <code>Aggregation</code> of your code by <code>Composition</code> according to the following examples:  <code>ProjectAggregation = ProjectComposition</code>  <code>IDeviceAggregation = IDeviceComposition</code>  <code>TagTableAggregation = TagTableComposition</code>  <code>CycleAggregation = CycleComposition</code>  <code>GraphicListAggregation = GraphicListComposition</code>  <code>TextListAggregation = TextListComposition</code>  <code>ConnectionAggregation = ConnectionComposition</code>  <code>MultiLingualGraphicAggregation = MultiLingualGraphicComposition</code>  <code>UpdateCheckResultMessageAggregation = UpdateCheckResultMessageComposition</code></li> <li>2. Compile the application.</li> </ol>
Folders have been replaced by groups in every relationship except HMI devices.	<ol style="list-style-type: none"> <li>1. Replace every <code>Folder</code> in your program code by <code>Group</code> except code parts which concern to HMI devices.</li> <li>2. Compile the application.</li> </ol>
The <code>GetAttributeNames</code> method has been replaced by the <code>GetAttributeInfos</code> method.	<ol style="list-style-type: none"> <li>1. Use <code>IList&lt;EngineeringAttributeInfo&gt; IEngineeringObject.GetAttributeInfos(AttributeAccessMode attributeAccessMode);</code> to determinate attributes.</li> <li>2. Compile the application. For more detailed information, refer to <code>Determining the object structure and attributes</code> (Page 109).</li> </ol>
The <code>Close</code> method for closing an object has changed.	<ol style="list-style-type: none"> <li>1. Replace <code>project.Close(CloseMode.PromptIfModified);</code> by <code>project.Close();</code></li> <li>2. Compile the application. For more detailed information, refer to <code>Closing a project</code> (Page 120).</li> </ol>

Change	Required program code adjustment
<p>Simultaneous access has been replaced by exclusive access and transactions.</p>	<ol style="list-style-type: none"> <li>1. Replace simultaneous access by exclusive access and transactions according to the following examples: <ul style="list-style-type: none"> <li>- TIA Portal Openness V13 SP1(obsolete): <pre>tiaProject.StartTransaction("Reseting project to default"); ... tiaProject.CommitTransaction();</pre> </li> <li>- TIA Portal Openness V14: <pre>//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();</pre> </li> </ul> </li> <li>2. Compile the application. See Exclusive access (Page 89) and Transaction handling (Page 91) for further information.</li> </ol>
<p>Online access to the CPU has been changed</p>	<ol style="list-style-type: none"> <li>1. Change the online access to the CPU according to the following examples: <ul style="list-style-type: none"> <li>- TIA Portal Openness V13 SP1(obsolete): <pre>((IOOnline)controllerTarget).GoOffline();</pre> </li> <li>- TIA Portal Openness V14: <pre>((DeviceItem)plcSoftware.Parent.Parent).GetService&lt;OnlineProvider&gt;().GoOffline();</pre> </li> </ul> </li> <li>2. Compile the application.</li> </ol>
<p>The hardware configuration has been changed</p>	<ol style="list-style-type: none"> <li>1. Change the hardware configuration: <pre>Device.Elements = Device.Items</pre> </li> <li>2. Remove the following hardware attributes: <ul style="list-style-type: none"> <li>- Device.InternalDeviceItem</li> <li>- Device.SubType</li> </ul> </li> <li>3. Compile the application.</li> </ol>

**See also**

- Handling exceptions (Page 407)
- What's new in TIA Portal Openness? (Page 23)
- Connecting to the TIA Portal (Page 74)



## 9.3.4 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 exception 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

"

- "Devices & networks" editor
  - Open, 164
- "Tags" editor
  - Starting, 359

## A

- Accessing
  - Master copy in project library, 149
- Acknowledging system events program-controlled, 82

## B

- Basic structure of an AML export file, 548
- Basic structure of an export file, 427, 552
- Block
  - Creating group, 308
  - Deleting, 308
  - Deleting group, 309
  - Exporting, 487
  - Generate source, 314
  - Importing, 535
  - Querying information, 304
- Block editor
  - Starting, 319

## C

- Compiling
  - Hardware, 117
  - Software, 117
  - Technology object, 325
  - Technology object group, 326
- Configuration
  - Your Openness application and the TIA Portal run on different computers, 45
- Connecting
  - Analog drives by data block, 337
  - Analog drives by hardware address, 335
  - Cam track, 352
  - Drives, 345
  - Encoders, 350
  - Encoders by data block, 338

- Encoders for analog drives by hardware address, 336
- Encoders for PROFIdrives by hardware address, 334
- Measuring input, 353
- Output cam, 352
- PROFIdrives by data block, 337
- PROFIdrives by hardware address, 333
- Synchronous axis with leading values, 354
- telegram 750, 348

### Connection to the TIA Portal

- Close, 83
- Setting up, 74

### Copy

- Content of a master copy in project folder, 152
- Master copy, 155

### Create

- User-defined folders for HMI tags, 252
- User-defined screen folders, 247
- User-defined script folders, 255

### Creating

- Cam track, 339
- Group for block, 308
- Measuring input, 339
- Output cam, 339
- Technology object, 324
- User-defined folder for PLC tag tables, 362

## D

### Data types

- Technology object, 322

### Deleting

- All screens, 249
- Block, 308
- Connection, 252
- Cycle, 250
- Deleting a PLC tag table from a folder, 366
- Graphic list, 251
- Group for block, 309
- Individual tag in a PLC tag table, 368
- Individual tags of a tag table, 253
- PLC constants, 369
- Program block, 308
- Project graphics, 116
- Screen, 247
- Screen template, 248
- Tag table, 254
- Technology object, 325

- Text list, 250
  - User data type, 318
  - User-defined folder for PLC tag tables, 363
  - VB script from a folder, 255
- E**
- Editing situation
    - Your Openness application and the TIA Portal run on the same computer, 46
  - Enumerating
    - All tags of a tag table, 253
    - Blocks, 302
    - Device items, 230
    - Devices, 214, 217
    - Multilingual texts, 107, 112
    - Parameter of technology object, 328
    - PLC tag tables, 363
    - PLC tags, 367
    - System subfolders, 300
    - Technology object, 327
    - User-defined block folders, 301
    - User-defined folder for PLC tags, 361
  - Enumerating device items, 230
  - Enumerating devices, 214, 217
  - Enumerating multilingual texts, 107, 112
  - Establishing a connection to the TIA Portal, 74
  - Example program, 49
  - Exceptions
    - When accessing the TIA Portal via public APIs, 407
  - Export file
    - Basic structure, 427, 548, 552
    - Contents, 416
    - Structure of the XML file, 427, 552
  - Export/import
    - Application, 35
  - Exportable screen objects, 449
  - Exporting
    - Block, 487
    - Individual tag or constant from a PLC tag table, 540
    - User data type, 487
- F**
- Finding
    - Cam track, 339
    - Measuring input, 339
    - Output cam, 339
  - Parameter of technology object, 329
  - Technology object, 327
  - Folders
    - Deleting, 161
  - Functions, 49
    - Closing a project, 120
    - Creating a user-defined folder for PLC tag tables, 362
    - Creating user-defined folders for HMI tags, 252
    - Creating user-defined screen folders, 247
    - Creating user-defined script subfolders, 255
    - Deleting a connection, 252
    - Deleting a cycle, 250
    - Deleting a graphic list, 251
    - Deleting a PLC tag table, 366
    - Deleting a screen, 247
    - Deleting a screen template, 248
    - Deleting a tag from a PLC tag table, 368
    - Deleting a tag from a tag table, 253
    - Deleting a tag table, 254
    - Deleting a text list, 250
    - Deleting a user-defined folder for PLC tag tables, 363
    - Deleting a VB script from a folder, 255
    - Deleting all screens, 249
    - Deleting project graphics, 116
    - Determining the system folder, 299
    - Enumerating blocks, 302
    - Enumerating device items, 230
    - Enumerating devices, 214, 217
    - Enumerating multilingual texts, 107, 112
    - Enumerating PLC tag tables in folders, 363
    - Enumerating PLC tags, 367
    - Enumerating system subfolders, 300
    - Enumerating tags of an HMI tag table, 253
    - Enumerating user-defined block folders, 301
    - Enumerating user-defined folders for PLC tags, 361
    - Exporting a tag or constant from a PLC tag table, 540
    - General, 74, 82, 83
    - General TIA portal settings, 102
    - HMI, 247, 248, 249, 250, 251, 252, 253, 254, 255
    - Importing a tag into a PLC tag table, 541
    - Importing PLC tag tables, 539
    - Limitation to projects of TIA Portal V13, 97
    - Open project, 97
    - PLC, 299, 300, 301, 302, 304, 362, 363, 366, 368, 369, 539, 540, 541
    - PLC constants, 369
    - Projects, 97, 102, 107, 112, 116, 119, 120, 165, 214, 217, 230, 360, 361, 363, 364, 367

- Public API application example, 67
- Querying information from a PLC tag table, 364
- Querying PLC and HMI targets, 165
- Querying system folders for PLC tags, 360
- Querying the "Program blocks" folder, 299
- Querying the block author, 304
- Querying the block family, 304
- Querying the block name, 304
- Querying the block number, 304
- Querying the block title, 304
- Querying the block type, 304
- Querying the block version, 304
- Querying the consistency attribute of a block, 304
- Querying the time stamp of a block, 304
- Reading the time of the last changes to a PLC tag table, 366
- Saving a project, 119

## G

- General TIA portal settings, 102
- Generate
  - source from block, 314
  - source from user data type, 314
- Global library
  - Accessing, 126, 130
  - Accessing language settings, 128

## H

- Hardware
  - Compiling, 117
- Hierarchy of hardware objects of the object model, 64
- HMI tags of the "UDT" data type, 438

## I

- Import/Export
  - Advanced XML formats for export/import of text lists, 444
  - Also export default values, 416
  - Basics, 411
  - Data structure, 427, 552
  - Editing an XML file, 415
  - Export format, 413
  - Export scope, 415
  - Export settings, 415
  - Exportable objects, 411
  - Exportable screen objects, 449
  - Exporting a screen from a screen folder, 454

- Exporting a screen with a faceplate instance, 469
- Exporting a selected tag, 436
- Exporting a tag from a tag table, 436
- Exporting all graphics of a project, 420
- Exporting all screen templates, 460
- Exporting blocks with know-how protection, 495
- Exporting blocks without know-how protection, 486
- Exporting configuration data, 415
- Exporting connections, 447
- Exporting cycles, 430
- Exporting graphic lists, 446
- Exporting HMI tag tables, 432
- Exporting multilingual comments, 590, 594, 602, 604
- Exporting only modified values, 416
- Exporting permanent areas, 458
- Exporting PLC tag table, 538
- Exporting pop-up screens, 464
- Exporting screen templates, 461
- Exporting screens of an HMI device, 453
- Exporting slide-in screen, 467
- Exporting system blocks, 532
- Exporting tags, 600
- Exporting text lists, 442
- Exporting VB scripts, 439, 440
- Field of application, 413
- Graphics, 419
- HMI, 430, 431, 432, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 446, 448, 449, 453, 454, 456, 458, 459, 460, 461, 463, 464, 465, 467, 468, 469, 471, 538
- Importable objects, 411
- Importing a graphic list, 446
- Importing a screen including a faceplate instance, 471
- Importing an HMI tag into a tag table, 437
- Importing configuration data, 417
- Importing connections, 448
- Importing cycles, 431
- Importing graphics to a project, 421
- Importing multilingual comments, 590, 594, 602, 604
- Importing permanent areas, 459
- Importing pop-up screen, 465
- Importing screen templates, 463
- Importing screens to an HMI device, 456
- Importing slide-in screen, 468
- Importing tag table to a tag folder, 435
- Importing tags, 600
- Importing text list, 443
- Importing VB scripts, 441

- Objects of AML, 548
- PLC, 486, 495, 532
- Procedure for importing, 418
- Project data, 420, 421
- Restricting exports to modified values, 416
- Restrictions, 413
- Round trip devices and modules, 569
- Setting the import behavior by means of program codes, 417
- Special considerations for integrated HMI tags, 438
- Stable AML GUIDs, 569

#### Importing

- An individual tag into a PLC tag table, 541
- Block, 535
- PLC tag tables, 539
- User data type, 543

#### Installation

- Access authentication check, 28
- Adding users to the user group, 28
- Standard steps for accessing the TIA Portal, 33
- TIA Openness V13 add-on package, 27

Installing the add-on package, 27

#### Instances

- Determining type versions, 155

Integrated HMI tags, 438

## L

#### Library

- Accessing folders, 137
- Determining type versions of instances, 155
- Functions, 125

## M

#### Master copies

- Deleting, 161

#### Master copy

- Copy content to project folder, 152
- Copying, 155

## O

Object model, 51

#### Objects

- Exportable objects, 411
- Importable objects, 411

#### Open

- "Devices & networks" editor, 164

Opening a project, 97

## P

#### Parameter of technology object

- Enumerating, 328
- Finding, 329
- Reading, 330
- Writing, 331

#### Parameters

- Counting, 358
- Easy Motion Control, 358
- PID control, 357
- S7-1500 Motion Control, 341

#### PLC

- Comparing, 293
- Comparison with actual status, 293
- Determining status, 257
- Disconnecting an online connection, 297
- Establishing an online connection, 297

#### Program block

- Deleting, 308

Programming overview, 49

#### Project

- Close, 120
- Open, 97
- Querying HMI targets, 165
- Querying PLC targets, 165
- Querying the device type, 165
- Save, 119

#### Project library

- Accessing, 126, 130
- Accessing master copies, 149

Public API application example, 67

## Q

#### Querying

- Block author, 304
- Block family, 304
- Block name, 304
- Block number, 304
- Block title, 304
- Block type, 304
- Block version, 304
- Consistency attribute of a block, 304
- Finding the, 299
- Information from a PLC tag table, 364
- Information of block, 304
- Information of user data type, 304
- Program blocks folder, 299
- System folders for PLC tags, 360

Technology object, 323  
Time stamp of a block, 304

## R

Read  
Time of the last changes to a tag table, 366  
Reading  
Parameter of technology object, 330

## S

Saving a project, 119  
Siemens.Engineering, 40  
Siemens.Engineering.Hmi, 40  
Siemens.Engineering.Hmi.Communication, 40  
Siemens.Engineering.Hmi.Cycle, 40  
Siemens.Engineering.Hmi.Globalization, 40  
Siemens.Engineering.Hmi.RuntimeScripting, 40  
Siemens.Engineering.Hmi.Screen, 40  
Siemens.Engineering.Hmi.Tag, 40  
Siemens.Engineering.Hmi.TextGraphicList, 40  
Siemens.Engineering.HW, 40  
Siemens.Engineering.SW, 40  
Software  
Compiling, 117  
Special considerations for HMI tags of the "UDT" data type, 438  
Starting  
"Tags" editor, 359  
Block editor, 319  
Status (PLC)  
Determining, 257  
Structure of the export data, 427, 548, 552

## T

Technology object, 320  
Compiling, 325  
Creating, 324  
Data types, 322  
Deleting, 325  
Enumerating, 327  
Finding, 327  
Querying, 323  
Technology object group  
Compiling, 326  
Terminating the connection to the TIA Portal, 83  
TIA Portal Openness, 43  
Access, 34  
Access rights, 28

Adding users to the user group, 28  
Basic concepts of aggregations, 171  
Basic concepts of associations, 170  
Basic concepts of object equality verification, 172  
Basic concepts when handling exceptions, 407  
Configuration, 45  
Export/import, 35  
Functional scope, 43  
Functions, 49  
Introduction, 43  
Necessary user knowledge, 25  
Programming overview, 49  
Public API, 49  
Requirements, 25  
Standard steps for accessing the TIA Portal, 33  
Typical tasks, 34

## Types

Deleting, 161

## U

User data type  
Deleting, 318  
Exporting, 487  
Generate source, 314  
Importing, 543  
Querying information, 304

## W

Writing  
Parameter of technology object, 331

## X

XML file  
Edit, 415  
Export, 416

