TIA Scripting Python

Using Python scripts to automate TIA Portal processes.

https://support.industry.siemens.com/cs/ww/en/view/109742322

Siemens Industry Online Support

Legal information

Use of application examples

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

Disclaimer of liability

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

Other information

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

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

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

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under https://www.siemens.com/cert.

Table of contents

1 TIA Scripting Python

TIA Scripting Python is a module for writing Python scripts to interact with the TIA Openness interface. It is designed to automate simple tasks in TIA Portal projects using a simple scripting language, and thus avoiding the need for complicated programming. It helps simplify tasks such as importing and compiling program blocks, exporting data, and comparing projects. This module simplifies some of the basic tasks of TIA Portal and makes it easier for users without much experience to work with TIA Openness. In addition, the installed TIA Portal versions are automatically recognized and used.

1.1 TIA Portal programming interface

The TIA Portal Openness API (Application Programming Interface) provided by TIA Portal allows you to automate recurring steps in your projects. This is useful as manual adjustments in projects involve a high susceptibility to errors. Moreover, this automation allows you to save time, enabling you to work more efficiently.

1.2 Prerequisites

1.2.1 Installed software

In order to run this program, the following software has to be installed:

- TIA Portal V15.1 (or newer)
- TIA Portal Openness V15.1 (or newer)

The requirements for the mentioned programs can be found in their respective documentations.

To work with TIA Scripting Python, Python is required.

- Python 3.12.X
- any Python IDE, e.g. Visual Studio Code

1.3 Python environment

1.3.1 Python installation

To check if you have Python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
python --version
```

If no Python version is installed, you can download it from https://www.python.org/. Python version 3.12.X is required to use.

1.3.2 Python Quickstart

Python is an interpreted programming language, which means that as developers write Python (.py) files in a text editor and then put those files into the Python interpreter to be executed. The way to run a Python file is like this on the command line:

```
python helloworld.py
```

Where “helloworld.py” is the name of your Python file.

1.4 Managing rights of TIA Portal Openness

1.4.1 Add user to Siemens TIA Openness group

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.

Follow the instructions within the Openness manual for detailed instructions.

1.4.2 Openness security-dialog

When you use the application the first time and it connects to TIA Portal via TIA Portal Openness, TIA Portal will prompt you to accept or reject the connection via a security dialog:

- 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.
- To create a whitelist entry for your TIA Portal Openness application follow these steps:
  - Click “Yes to all” at the prompt to display an User Account Control Dialog.
  - 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.

Further information can be found in the Openness manual.

1.5 How to use TIA Scripting Python

There are two possibilities to use TIA Scripting Python.

1.5.1 Use TIA Scripting Python via file import

Unzip TiaScriptingPython_x.x.x.zip to a folder on your hard-drive. To make this folder known during execution of scripts, set a global environment variable “TIA_SCRIPTING”, whereas the value is the path of the previously unzipped TIA Scripting Python binaries. This way, “TIA_SCRIPTING” will be used to find the Python module of the TIA Scripting Python by Python scripts.

1.5.2 Use TIA Scripting Python as Python package

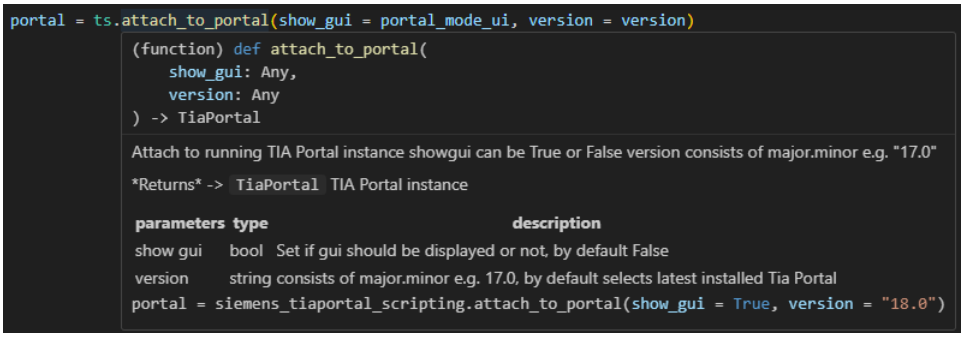
You can install TIA Scripting Python as Python package. Install the package with pip (Python package manager):

```
pip install siemens_tiaportal_scripting-x.x.x-cp312-cp312-win_amd64.whl
```

This will install TIA Scripting Python as Python package default at users Python site packages folders.

Intellisense Support

Intellisense support can be used only if you install TIA Scripting Python as Python package. Pylance language server (Python Intellisense) is able to provide full Intellisense Support. Pylance is default by Visual Studio Code Python Plugin and Microsoft Visual Studio Python. Other Language servers like “Jedi” are currently untested.



1.6 Scripts

The folder “Scripts” contains example scripts which are using TIA Scripting Python to interact with TIA Portal.

Script	Description
attacher.py	Attaches to the currently running TIA Portal instance (with user-interface) and gets the opened project. It prints the names of all available PLCs and opens the device editor of the first PLC.
exporter.py	Enumerates all software-elements of a PLC and exports them to the configured folder.
importer.py	Imports all exported elements to a PLC.
upgrader.py	Upgrades the project to a higher version of TIA Portal. The process compiles the hardware and software. Hardware can also be upgraded to newest firmware version.

1.7 Create an Executable from your script

It is possible to create standalone executable files from .py scripts. First the package “pyinstaller” has to be install:

```
pip install pyinstaller
```

After pyinstaller is installed:

```
pyinstaller.exe --onefile demo.py -p D:\PyLibs\
```

When building a Python executable, it's important to ensure that local binaries (such as `siemens_tia_scripting.pyd`) are used instead of any globally installed packages. The following instructions will guide you through the process of achieving this using **PyInstaller**.

To ensure your Python script loads local binaries, include the following code. This will dynamically locate the `.pyd` file bundled with the executable:

```
import importlib.util
import os
import sys

# Get the file from the location extracted by PyInstaller
def get_file_path(file_name):
    if hasattr(sys, '_MEIPASS'):
        return os.path.join(sys._MEIPASS, file_name)
    return os.path.join(os.path.abspath("."), file_name)

# Import this specific package, bypassing globally installed ones
spec = importlib.util.spec_from_file_location("siemens_tia_scripting",
get_file_path("siemens_tia_scripting.pyd"))
ts = importlib.util.module_from_spec(spec)
spec.loader.exec_module(ts)
```

`get_file_path()`: This function ensures that the correct local version of the binary file is used, especially when the script is packaged into an executable.
The code dynamically loads the `.pyd` binary, ensuring that local binaries are used, bypassing any globally installed packages.

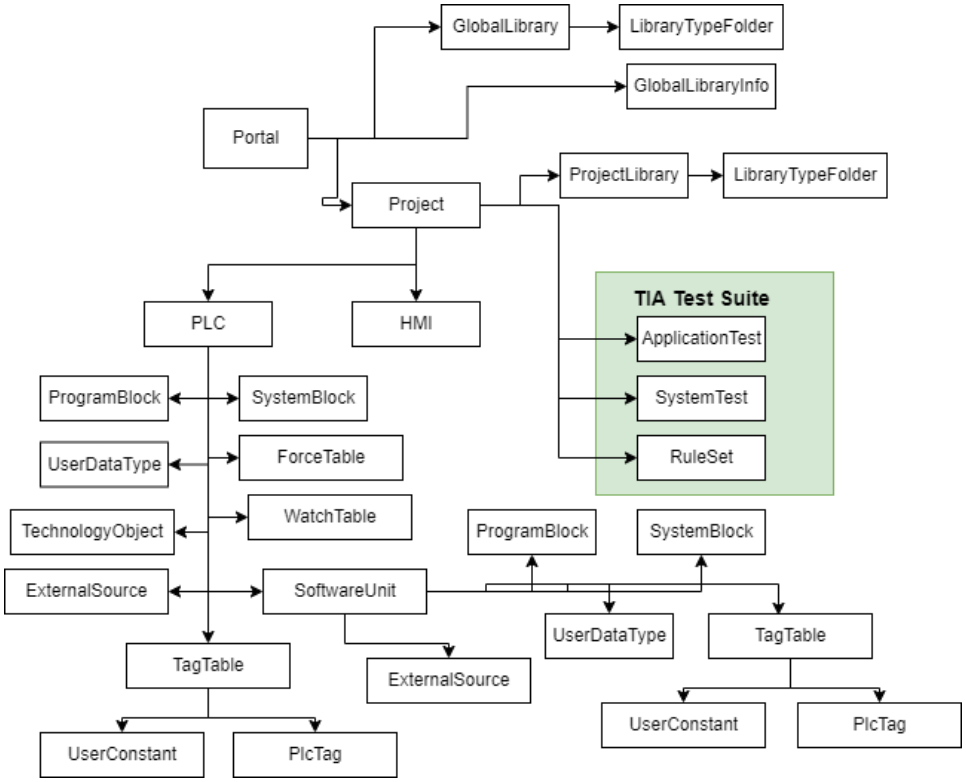
you can use the following PyInstaller command to build the executable directly from the command line:

```
pyinstaller -F /mytiascript.py --workpath /temp -n MyTiaExecutable --clean --distpath
./output --add-data "../dep/:."
```

- **-F**: Creates a single standalone executable.
- **/mytiascript.py**: The path to your Python script that needs to be compiled.
- –**workpath /temp**: Specifies the temporary working directory during the build process (in this case, `../temp`).
- **-n MyTiaExecutable**: Specifies the name of the resulting executable.
- –**clean**: Cleans up any temporary files from previous builds before starting.
- –**distpath ./output**: The path where the final executable will be placed (in this case, `./output`).
- –**add-data "../dep/:."**: Adds additional data (like `.pyd` binaries) required by the executable (in this case, the contents of the `../dep/` directory).

1.8 Hierarchy

The overview shows the available models and their dependency to each other. Accessing a program block from a PLC requires that a TIA Portal connection is established, a project opened and a controller object selected.



2 Function List

The first lines of any script should import the TIA Scripting Python, whereas the previously defined system environment variable is used therefore.

```
sys.path.append(os.getenv('TIA_SCRIPTING'))
import siemens_tia_scripting
```

If the import was successful, the appropriate methods from TIA Scripting Python can be used.

2.1 Devices - Hmi

The class represents HMI (Human Machine Interface) devices in TIA Portal. It allows controlling and interacting with the HMI, including compilation and upgrading tasks.

2.1.1 get_name()

Get the name of the HMI

Returns → `str` Name of the HMI

```
hmi_name = hmi.get_name()
```

2.1.2 open_device_editor()

Open the editor of the HMI-device in TIA Portal

```
hmi.open_device_editor()
```

2.1.3 compile_hardware()

Compile the hardware of the HMI

Returns → `bool` Result of the compile

Returns true if compile has errors, otherwise returns false

```
result = hmi.compile_hardware()
```

2.1.4 compile_software()

Compile the software of the HMI

Returns → `bool` Result of the compile

Returns true if compile has errors, otherwise returns false

```
result = hmi.compile_software()
```

2.1.5 upgrade_hardware(full_upgrade: bool)

Update hardware of the HMI

If `full_upgrade` is set to True, all devices will be changed to the newest available order number (Device type) and firmware:

from CPU1511F 6ES7 511-1FK0**0**-0AB0 **V1.7** to 6ES7 511-1FK0**0**-0AB0 **V1.8**.

With full upgrade: from CPU1511F 6ES7 511-1FK0**0**-0AB0 **V1.7** to 6ES7 511-1FL0**3**-0AB0 **V3.0**.

Parameters	Type	Description
full_upgrade	bool	Set if for full upgrade (latest order number and firmware version)

```
hmi.upgrade_hardware(full_upgrade = True)
```

2.1.6 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.1.7 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.2 Devices - Plc

This represents a PLC (Programmable Logical Controller). It allows for controlling and interacting with the PLC, including downloading, comparing online and offline states, and retrieving information about PLC elements.

2.2.1 get_name()

Get the name of the PLC

Returns → `str` Name of the PLC

```
plc_name = plc.get_name()
```

2.2.2 open_device_editor()

Open the editor of the PLC-device in TIA Portal

```
plc.open_device_editor()
```

2.2.3 get_online_state()

Get the current online state of the PLC

Returns → `str` Online state

```
plc.get_online_state()
```

2.2.4 go_offline()

Go offline with the PLC

```
plc.go_offline()
```

2.2.5 go_online(mode: str, pci_interface: str, ip_address: str)

Go online with the PLC

Returns → `str` Online status

Parameters	Type	Description
mode	str	Configuration mode
pci_interface	str	PC Interface name
ip_address	str	IP address

```
plc.go_online(mode = "PN/IE", pci_interface = "PLCSIM", ip_address = "192.168.0.10")
```

2.2.6 download(mode: str, pci_interface: str, ip_address: str)

Download the PLC

Returns → `str` Result of the download

Parameters	Type	Description
mode	str	Configuration mode
pci_interface	str	PC Interface name
ip_address	str	IP address

```
result = plc.download(mode = "PN/IE", pci_interface = "PLCSIM", ip_address =
"192.168.0.10")
```

2.2.7 download_to_memory_card(download_directory: str)

Download the PLC-configuration to memory card

Returns → `str` Result of the download

Parameters	Type	Description
download_directory	str	Download directory

```
result = plc.download_to_memory_card(download_directory = "E:\\")
```

2.2.8 get_plc_tag_tables(folder_path: Optional[str] = None)

Get the list of the PLC Tag Tables

Returns → `List[PlcTagTable]` List of the PLC Tag tables

Parameters	Type	Description
folder_path	str	The group path of the item

```
plc_tag_tables = plc.get_plc_tag_tables(folder_path = "group1/group2")
```

2.2.9 update_module_description()

Update the module description of the PLC

Returns → `bool` Result state

```
result = plc.update_module_description()
```

2.2.10 compile_hardware()

Compile the hardware of the PLC

Returns → `bool` Result of the compile

Returns true if compile has errors, otherwise returns false

```
result = plc.compile_hardware()
```

2.2.11 compile_software()

Compile the software of the PLC

Returns → `bool` Result of the compile

Returns true if compile has errors, otherwise returns false

```
result = plc.compile_software()
```

2.2.12 upgrade_hardware(full_upgrade: bool)

Upgrade the hardware the PLC

If `full_upgrade` is set to True, all devices will be changed to the newest available order number (Device type) and firmware:

from CPU1511F 6ES7 511-1FK0**0**-0AB0 **V1.7** to 6ES7 511-1FK0**0**-0AB0 **V1.8**.

With full upgrade: from CPU1511F 6ES7 511-1FK0**0**-0AB0 **V1.7** to 6ES7 511-1FL0**3**-0AB0 **V3.0**.

Parameters	Type	Description
full_upgrade	bool	Set if for full upgrade (latest order number and firmware version)

```
plc.upgrade_hardware(full_upgrade = True)
```

2.2.13 compare_to_online()

Compare actual PLC state to the online state

Returns → `bool` Result of the compare

Returns false if folders are Identical, otherwise returns true

```
result = plc.compare_to_online()
```

2.2.14 get_program_blocks(folder_path: Optional[str] = None)

Get the list of program blocks

Returns → `List[ProgramBlock]` List of the program blocks

Parameters	Type	Description
folder_path	str	The group path of the item

```
blocks = plc.get_program_blocks(folder_path = "group1/group2")
```

2.2.15 get_system_blocks()

Get the list of system blocks

Returns → `List[SystemBlock]` List of the system blocks

```
blocks = plc.get_system_blocks()
```

2.2.16 get_user_data_types(folder_path: Optional[str] = None)

Get the list of PLC data types

Returns → `List[UserDataType]` List of the PLC data types

Parameters	Type	Description
folder_path	str	The group path of the item

```
udts = plc.get_user_data_types(folder_path = "group1/group2")
```

2.2.17 get_external_sources(folder_path: Optional[str] = None)

Get the list of external source files

Returns → `List[ExternalSource]` List of the external source files of the PLC

Parameters	Type	Description
folder_path	str	The group path of the item

```
ext_sources = plc.get_external_sources()
```

2.2.18 get_force_tables()

Get the list of force tables

Returns → `List[ForceTable]` List of the force tables of the PLC

```
tables = plc.get_force_tables()
```

2.2.19 get_watch_tables(folder_path: Optional[str] = None)

Get the list of watch tables

Returns → `List[WatchTable]` List of the watch tables of the PLC

Parameters	Type	Description
folder_path	str	The group path of the item

```
tables = plc.get_watch_tables()
```

2.2.20 get_technology_objects(folder_path: Optional[str] = None)

Get the list of technology objects

Returns → `List[TechnologyObject]` List of the technology objects of the PLC

Parameters	Type	Description
folder_path	str	The group path of the item

```
tos = plc.get_technology_objects()
```

2.2.21 get_software_units()

Get the list of software units

Returns → `List[SoftwareUnit]` List of the software units

```
software_units = plc.get_software_units()
```

2.2.22 get_safety_administration()

Get Safety administration of the PLC

Returns → `SafetyAdministration` Safety Administration of the PLC

```
sa = plc.get_safety_administration()
```

2.2.23 import_blocks(import_root_directory: str)

Import program blocks from a directory to the PLC

Parameters	Type	Description
import_root_directory	str	Directory of the import folder

```
plc.import_blocks(import_root_directory = "C:\\ws\\importfolder\\PLC_1\\Program blocks")
```

2.2.24 import_plc_tags(import_root_directory: str)

Import tags from a directory to the PLC

Parameters	Type	Description
import_root_directory	str	Directory of the import folder

```
plc.import_plc_tags(import_root_directory = "C:\\ws\\importfolder\\PLC_1\\PLC tags")
```

2.2.25 import_data_types(import_root_directory: str)

Import user data types from a directory to the PLC

Parameters	Type	Description
import_root_directory	str	Directory of the import folder

```
plc.import_data_types(import_root_directory = "C:\\ws\\importfolder\\PLC_1\\PLC data types")
```

2.2.26 import_technology_objects(import_root_directory: str)

Import technology objects from a directory to the PLC

Parameters	Type	Description
import_root_directory	str	Directory of the import folder

```
plc.import_technology_objects(import_root_directory = "C:\\ws\\importfolder\\PLC_1\\Technology objects")
```

2.2.27 import_watch_tables(import_root_directory: str)

Import watch tables from a directory to the PLC

Parameters	Type	Description
import_root_directory	str	Directory of the import folder

```
plc.import_watch_tables(import_root_directory = "C:\\ws\\importfolder\\PLC_1\\Watch and force tables")
```

2.2.28 import_software_units(import_root_directory: str)

Import software units from a directory to the PLC

Parameters	Type	Description
import_root_directory	str	Directory of the import folder

```
plc.import_software_units(import_root_directory =
"C:\\ws\\importfolder\\PLC_1\\Software Units")
```

2.2.29 safety_print(print_file: str)

Create a safety printout of the PLC and print it to a file

If the file already exists, it is overwritten.

Returns → `bool` Returns true on success

Parameters	Type	Description
print_file	str	Full path of the printout file

```
plc.safety_print(print_file = "C:\\ws\\safetyprint\\F_PLC_Printout.pdf")
```

2.2.30 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.2.31 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.3 Enums

2.3.1 PortalMode

enum element	value
WithGraphicalUserInterface	0
WithoutGraphicalUserInterface	1
AnyUserInterface	2

2.3.2 UmacUserMode

enum element	value
Project	0
Global	1

2.3.3 ExportFormats

enum element	value
SimaticML	0
ExternalSource	1
SimaticSD	2

2.3.4 ExportOptions

enum element	value
WithDefaults	0
Nan	1
WithReadOnly	2

2.4 Global - Global

Functions to start or attach to TIA Portal are available to get the TIA Portal instance.

2.4.1 open_portal(portal_mode: Optional[Enums.PortalMode] = None, version: Optional[str] = None)

Open a new TIA Portal instance Version-string in format major.minor e.g. "18.0"

Returns → `Portal` TIA Portal instance

Parameters	Type	Description
portal_mode	Enums.PortalMode	With or without user-interface
version	str	Version string of TIA Portal to be used in format major.minor e.g. 18.0, (Latest installed TIA Portal version by default)

Example usage

```
portal = siemens_tia_scripting.open_portal(portal_mode =
siemens_tia_scripting.Enums.PortalMode.WithGraphicalUserInterface, version = "18.0")
```

2.4.2 attach_portal(portal_mode: Optional[Enums.PortalMode] = None, version: Optional[str] = None)

Attach to running TIA Portal instance Version-string in format major.minor e.g. "18.0"

Returns → `Portal` TIA Portal instance

Parameters	Type	Description
portal_mode	Enums.PortalMode	With or without user-interface
version	str	Version string of TIA Portal to be used in format major.minor e.g. 18.0, (Latest installed TIA Portal version by default)

```
portal = siemens_tia_scripting.attach_portal(portal_mode =
Enums.PortalMode.WithGraphicalUserInterface, version = "18.0")
```

2.4.3 open_attach_project(project_file_path: str, portal_mode: Optional[Enums.PortalMode] = None)

Attach to running TIA Portal instance with already open project Or opens the project with a new instance of fitting TIA Portal version

Returns → `Project` TIA Project instance

Parameters	Type	Description
project_file_path	str	Full path of the project file
portal_mode	Enums.PortalMode	With or without user-interface

```
project = siemens_tia_scripting.open_attach_project(project_file_path =
"C:\\ws\\testproj\\testproj.ap17", portal_mode =
siemens_tia_scripting.Enums.PortalMode.WithGraphicalUserInterface)
```

2.4.4 get_installed_bundles()

Get a list of installed TIA Portal bundles

Returns → `List[ProductBundle]` List of installed bundles

```
product_bundles = siemens_tia_scripting.get_installed_bundles()
```

2.4.5 get_installed_products()

Get a list of installed TIA Portal products

Returns → `List[Product]` List of installed products

```
products = siemens_tia_scripting.get_installed_products()
```

2.4.6 set_umac_credentials(user_name: str, user_password: str, user_type: Enums.UmacUserMode)

Set UMAC Credentials which will be used for protected libraries or projects

Parameters	Type	Description
user_name	str	User name
user_password	str	Password of the user
user_type	Enums.UmacUserMode	Project or global user

```
siemens_tia_scripting.set_umac_credentials(user_name = "admin", user_password =
"Password123", user_type = Enums.UmacUserMode.Project)
```

2.4.7 set_logging(path: str, console: bool)

Set logging output path or console (stdout) output

Parameters	Type	Description
path	str	Fullpath of the log file
console	bool	(optional) True for Enable console output, False to disable the standard output to console window

```
siemens_tia_scripting.set_logging(path = "c:\\ws\\tiascripting.log", console = False)
```

2.5 Global - Portal

The Portal-object represents an open TIA Portal instance and provides functionality to manage projects and libraries within TIA Portal.

2.5.1 get_process_id()

Get the TIA Portal process ID

Returns → `int` TIA Portal process id

```
id = portal.get_process_id()
```

2.5.2 open_project(project_file_path: str, server_project_view: Optional[bool] = None)

Open a TIA Portal project or local session - optional in server view

Returns → `Project` TIA Portal project

Parameters	Type	Description
project_file_path	str	Full path of the project file
server_project_view	str	Default: false. If parameter set to true, open in server project view

```
project = portal.open_project(project_file_path = "C:\\ws\\testproj\\testproj.ap17")
```

2.5.3 open_project_with_copy(project_file_path: str, target_directory_path: str, delete_existing_project: bool)

Open a TIA Portal project or local session in separate folder

Returns → `Project` TIA Portal project

Parameters	Type	Description
project_file_path	str	Full path of the project file
target_directory_path	str	Temporary path where the project should be copied before opened
delete_existing_project	bool	Defines if the temporary project should be deleted

```
project = portal.open_project_with_copy(project_file_path =
"C:\\ws\\testproj\\testproj.ap17", target_directory_path = "C:\\ws\\temp" ,
delete_existing_project = True)
```

2.5.4 retrieve_archive(target_directory_path: str, archive_file_path: str, delete_existing_project: bool)

Retrieve a TIA Portal project archive

Returns → `Project` TIA Portal project

Parameters	Type	Description
target_directory_path	str	Temporary path where project should be copied before opened
archive_file_path	str	Full path of the archived project file
delete_existing_project	bool	Defines if the temporary project should be deleted

```
project = portal.retrieve_archive(archive_file_path =
"C:\\ws\\testproj\\testproj.zap17", target_directory_path = "C:\\ws\\temp" ,
delete_existing_project = True)
```

2.5.5 create_project(target_directory_path: str, project_name: str, delete_existing_project: bool)

Create a new TIA Portal project

Returns → `Project` TIA Portal project

Parameters	Type	Description
target_directory_path	str	Temporary path where project should be copied before opened
project_name	str	Name of the new project
delete_existing_project	bool	Defines if the temporary project should be deleted

```
project = portal.create_project(target_directory_path = "C:\\ws\\temp" , project_name
= "MyNewProject" delete_existing_project = True)
```

2.5.6 get_project()

Get opened TIA Portal project of the current TIA Portal instance

Returns → `Project` TIA Portal project

```
project = portal.get_project()
```

2.5.7 get_global_library(library_name: str)

Get global library by name

Returns → `GlobalLibrary` Global Library

Parameters	Type	Description
library_name	str	Name of the global library

```
global_lib = portal.get_global_library(library_name = "GlobalLib1")
```

2.5.8 open_global_library(library_path: str)

Open global library by path

Returns → `GlobalLibrary` Global Library

Parameters	Type	Description
library_path	str	Full path of the global library

```
global_lib = portal.open_global_library(library_path =
"C:\\ws\\testlib\\testlib.al17")
```

2.5.9 open_global_library_with_copy(target_directory_path: str, library_path: str, delete_existing_project: bool)

Open global library from a copy

Returns → `GlobalLibrary` Global Library

Parameters	Type	Description
target_directory_path	str	Temporary path where the library should be copied before opened
library_path	str	Full path of the global library
delete_existing_project	bool	Defines if the temporary project should be deleted

```
global_lib = portal.open_global_library_with_copy(target_directory_path =
"C:\\ws\\temp", library_path = "C:\\ws\\testlib\\testlib.al17",
delete_existing_project = True)
```

2.5.10 retrieve_archive_library(target_directory_path: str, archive_file_path: str, delete_existing_project: bool)

Retrieve a TIA Portal library archive

Returns → `GlobalLibrary` Global Library

Parameters	Type	Description
target_directory_path	str	Temporary path where library should be copied before opened
archive_file_path	str	Full path of the archived library file
delete_existing_project	bool	Defines if the temporary project should be deleted

```
global_lib = portal.retrieve_archive_library(target_directory_path = "C:\\ws\\temp",
archive_file_path = "C:\\ws\\testproj\\testlib.zal17", delete_existing_project =
True)
```

2.5.11 close_portal()

Close the TIA Portal instance

```
portal.close_portal()
```

2.5.12 close_global_library(global_library_name: str)

Close the global library instance

Parameters	Type	Description
global_library_name	str	Name of the global library

```
portal.close_global_library(global_library_name = "GlobalLib1")
```

2.5.13 get_project_servers(host_filter: Optional[str] = None)

Get a list of TIA Project-Servers found by host-filter

Returns → `List[ProjectServer]` List of TIA Project-Servers

Parameters	Type	Description
host_filter	str	Host of the TIA Project-Servers

```
portal.get_project_servers(host_filter = "CompanyServer")
```

2.5.14 get_project_server(url: str)

Get the TIA Project-Server found by URL

Returns → `ProjectServer` TIA Project-Server

Parameters	Type	Description
url	str	URL of the TIA Project-Server

```
portal.get_project_server(url = "https://project.server.net:8735/")
```

2.5.15 detach()

Detach from the TIA Portal instance

```
portal.detach()
```

2.6 Global - Product

The Product-object represents an object of installed Siemens product software.

2.6.1 get_name()

Get the name of the TIA Portal product

Returns → `str` Name of the TIA Portal product

```
product_name = product.get_name()
```

2.6.2 get_release()

Get the release of the TIA Portal product

Returns → `str` Release tag of the TIA Portal product

```
product_release = product.get_release()
```

2.6.3 get_version()

Get the version of the TIA Portal product

Returns → `str` Version of the TIA Portal product

```
product_version = product.get_version()
```

2.7 Global - ProductBundle

The ProductBundle-object represents an object of installed Siemens bundle software.

2.7.1 get_title()

Get the title of the TIA Portal Bundle

Returns → `str` Title of the Siemens bundle

```
bundle_title = bundle.get_title()
```

2.7.2 get_release()

Get the release of the TIA Portal Bundle

Returns → `str` Release of the Siemens bundle

```
bundle_release = bundle.get_release()
```

2.7.3 get_products()

Get the list of products of the TIA Portal Bundle

Returns → `List[Product]` List of Siemens products

```
bundle_products = bundle.get_products()
```

2.8 Library - GlobalLibrary

The class provides methods to interact with TIA Portal global libraries. It allows you to get information about the library, save it, get the author, check if it's modified and read-only state, and perform various operations on the library.

2.8.1 get_name()

Get the name of the global library

Returns → `str` Name of the global library

```
name = global_lib.get_name()
```

2.8.2 save()

Save the global library

```
global_lib.save()
```

2.8.3 get_author()

Get author of the global library

Returns → `str` Name of the author

```
author = global_lib.get_author()
```

2.8.4 get_path()

Get the path of the global library

Returns → `str` Full path of the library

```
path = global_lib.get_path()
```

2.8.5 get_library_type_folder()

Get the folder containing library types & library type folders

Returns → `LibraryTypeFolder` Library type folder

```
library_type = global_lib.get_library_type_folder()
```

2.8.6 is_modified()

Check if the global library was modified

Returns → `bool` State if library was modified

```
state = global_lib.is_modified()
```

2.8.7 is_read_only()

Check if the global library is readonly

Returns → `bool` State if global library is readonly

```
state = global_lib.is_read_only()
```

2.8.8 update_library(update_mode: int, delete_mode: int, conflict_mode: int, typename: Optional[str] = None, library_name: Optional[str] = None)

Update target library with type(s) from source

Parameters	Type	Description
type_name	str	This option should be used when a specific type has to be updated
library_name	str	This option should be used when a specific global library has to be updated
update_mode	integer (enum)	[1: 'ForceSetAnyUpdatedVersionAsDefault', 2: 'NoDefaultVersionChange', 3: 'SetOnlyHigherUpdatedVersionAsDefault']
delete_mode	integer (enum)	[0: 'AutomaticallyDelete', 1: 'DoNotDelete']
conflict_mode	integer (enum)	[1: 'CancelIfStructureConflicts', 2: 'RetainStructure', 3: 'UpdateStructure']

```
global_lib.update_library(update_mode = 1, delete_mode = 1, conflict_mode = 3)
```

2.8.9 update_project(update_mode: int, delete_mode: int, conflict_mode: int)

Update current project with type(s) from global library

Parameters	Type	Description
update_mode	integer (enum)	[1: 'ForceSetAnyUpdatedVersionAsDefault', 2: 'NoDefaultVersionChange', 3: 'SetOnlyHigherUpdatedVersionAsDefault']
delete_mode	integer (enum)	[0: 'AutomaticallyDelete', 1: 'DoNotDelete']
conflict_mode	integer (enum)	[1: 'CancelIfStructureConflicts', 2: 'RetainStructure', 3: 'UpdateStructure']

```
global_lib.update_project(update_mode = 1, delete_mode = 1, conflict_mode = 3)
```

2.8.10 archive(target_directory_path: str, archive_name: str, delete_existing_archive: bool)

Archive the TIA Portal library

Parameters	Type	Description
target_directory_path	str	Folder path where archive should be placed
archive_name	str	Name of the archive file
delete_existing_archive	bool	Set if the output file should be deleted first

```
global_lib.archive(target_directory_path = "C:\\ws\\newfolder", archive_name = "testglobalib" , delete_existing_archive = True)
```

2.8.11 get_property(name: str)

Get the property of the object
Properties which are not string will be converted to string if possible.
Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.8.12 get_properties()

Get all properties of the object
Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.9 Library - GlobalLibraryInfo

The class represents information about a global library in the context of TIA Portal.

2.9.1 get_name()

Get the name of the global library info
Returns → `str` Name of the global library info

```
name = global_lib_info.get_name()
```

2.9.2 get_property(name: str)

Get the property of the object
Properties which are not string will be converted to string if possible.
Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.9.3 get_properties()

Get all properties of the object
Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.10 Library - LibraryType

The class represents a library type in the context of TIA Portal.

2.10.1 get_name()

Get the name of the library type
Returns → `str` Name of the type

```
name = lib_type.get_name()
```

2.10.2 get_author()

Get the author of the library type
Returns → `str` Author of the type

```
author = lib_type.get_author()
```

2.10.3 get_guid()

Get the guid of the library type
Returns → `str` Guid of the type

```
guid = lib_type.get_guid()
```

2.10.4 get_versions()

Get the versions of the library type
Returns → `List[LibraryTypeVersion]` Versions of the type

```
versions = lib_type.get_versions()
```

2.10.5 find_version(version: str)

Find the version of the library type
Returns → `LibraryTypeVersion` Version des Bibliothekstyp

```
typeversion = lib_type.find_version(version = "1.0.0")
```

2.10.6 get_property(name: str)

Get the property of the object
Properties which are not string will be converted to string if possible.
Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.10.7 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.11 Library - LibraryTypeFolder

The class represents a folder containing library types and subfolders in the context of TIA Portal.

2.11.1 get_name()

Get the name of the folder

Returns → `str` Name of the folder

```
name = lib_folder.get_name()
```

2.11.2 get_folders()

Get the library type folders

Returns → `List[LibraryTypeFolder]` List of folders

```
folders = lib_folder.get_folders()
```

2.11.3 get_types()

Get the library types of the folder

Returns → `List[LibraryType]` List of types

```
types = lib_folder.get_types()
```

2.11.4 find_library_type(library_type_name: str)

Find the library type by name

Returns → `LibraryType` Library type

Parameters	Type	Description
library_type_name	str	Name of the library type

```
type = lib_folder.find_library_type(library_type_name = "MyType")
```

2.11.5 find_folder(folder_name: str)

Find the library type folder from the subfolders

Returns → `LibraryTypeFolder` Library type folder

Parameters	Type	Description
folder_name	str	Name of the subfolder

```
folder = lib_folder.find_folder(folder_name = "MyFolder")
```

2.12 Library - LibraryTypeVersion

The class represents a version of a library type in the context of TIA Portal.

2.12.1 get_author()

Get the author of the library type version
Returns → `str` Author

```
author = projectlib.get_author()
```

2.12.2 get_guid()

Get the GUID of the library type version
Returns → `str` GUID

```
guid = lib_type_version.get_guid()
```

2.12.3 get_version_number()

Get the version number of the library type version
Returns → `str` version number

```
version_number = lib_type_version.get_version_number()
```

2.12.4 get_modified_date()

Get the modification date of the library type version
Returns → `str` Modification date

```
date = lib_type_version.get_modified_date()
```

2.12.5 get_state()

Get the state of the library type version
Returns → `str` State of the library type version

```
state = lib_type_version.get_state()
```

2.12.6 get_type_object()

Get the type of the library type version
Returns → `LibraryType` Library type object

```
lib_type = lib_type_version.get_type_object()
```

2.12.7 get_property(name: str)

Get the property of the object
Properties which are not string will be converted to string if possible.
Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.12.8 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.13 Library - ProjectLibrary

The class represents a project library in the context of TIA Portal.

2.13.1 get_type_folder()

Get the folder containing library types & library type folders

Returns → `LibraryTypeFolder` Library type folder

```
typefolder = projectlib.get_type_folder()
```

2.14 PLC Data - ExternalSource

The class represents an external source in TIA Portal. It provides methods for getting information about the external source, retrieving properties and generating blocks.

2.14.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.14.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.14.3 block_gen()

Generate the blocks from the external source

```
ext_source.block_gen()
```

2.14.4 delete()

Delete the object

```
tiap_object.delete()
```

2.14.5 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.15 PLC Data - ForceTable

The class represents a force table in TIA Portal. It provides methods for getting information about the force table, retrieving properties and exporting the table.

2.15.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.15.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.15.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.15.4 is_consistent()

Check if the force table is consistent

Returns → `bool` True if consistent

```
value = force_table.is_consistent()
```

2.15.5 show_in_editor()

Open the PLC object in the editor

```
plc_object.show_in_editor()
```

2.15.6 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.16 PLC Data - NamedValueType

The class represents a Named value data type (NVT) in TIA Portal. It provides methods for managing and controlling Named value data types, exporting the data type, retrieving data type information and checking cross-references.

2.16.1 get_name()

Get the name of the Named value data type

Returns → `str` Name of the NVT

```
name = nvt.get_name()
```

2.16.2 get_namespace()

Get the name of the Named value data type namespace

Returns → `str` Name of the namespace

```
name = nvt.get_namespace()
```

2.16.3 export(target_directory_path: str, keep_folder_structure: Optional[bool] = None)

Export the Named value data type

If `keep_folder_structure` is set to true, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
nvt.export(target_directory_path = "C:\\ws\\export")
```

2.17 PLC Data - PlcTag

The class provides methods to interact with TIA Portal PLC Tags. It allows you to get the name, properties and perform export operations on PLC Tags.

2.17.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.17.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```


2.17.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.17.4 delete()

Delete the object

```
tiap_object.delete()
```

2.17.5 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.18 PLC Data - PlcTagTable

The represents a table of PLC tags in TIA Portal. It provides methods for managing and controlling PLC tags, exporting the table, retrieving tags and user constants and checking cross-references.

2.18.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.18.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.18.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.18.4 get_plc_tags()

Get the list of PLC tags

Returns → `List[PlcTag]` List of the PLC tags

```
plctags = table.get_plc_tags()
```

2.18.5 get_user_constants()

Get the list of the user constants

Returns → `List[UserConstant]` List of the PLC constants

```
constants = table.get_user_constants()
```

2.18.6 export_cross_references(target_directory_path: str, filter: int)

Export cross references of the PLC tag table filter → integer - [1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

Parameters	Type	Description
target_directory_path	str	Folder path for the export
filter	integer (enum)	[1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

```
table.export_cross_references(target_directory_path =
"C:\\ws\\exportcrossreferences", filter = 2)
```

2.18.7 show_in_editor()

Open the PLC object in the editor

```
plc_object.show_in_editor()
```

2.18.8 delete()

Delete the object

```
tiap_object.delete()
```

2.18.9 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.19 PLC Data - ProgramBlock

This represents a program block in TIA Portal. It provides various methods for managing and controlling program blocks.

2.19.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.19.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.19.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.19.4 compile()

Compile the program block

```
programblock.compile()
```

2.19.5 is_consistent()

Check if the program block is consistent

Returns → `bool` True if consistent

```
value = programblock.is_consistent()
```

2.19.6 export_cross_references(target_directory_path: str, filter: int)

Export cross references of the program block filter → integer - [1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

Parameters	Type	Description
target_directory_path	str	Folder path for the export
filter	integer (enum)	[1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

```
programblock.export_cross_references(target_directory_path =
"C:\\ws\\exportcrossreferences", filter = 2)
```

2.19.7 show_in_editor()

Open the PLC object in the editor

```
plc_object.show_in_editor()
```

2.19.8 get_type_version_guid()

Get the GUID of the type version

Returns → `str` GUID

```
guid = plc_object.get_type_version_guid()
```

2.19.9 get_type_guid()

Get the GUID of the type

Returns → `str` GUID

```
guid = plc_object.get_type_guid()
```

2.19.10 delete()

Delete the object

```
tiap_object.delete()
```

2.19.11 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.20 PLC Data - SafetyAdministration

The class provides methods to interact with TIA Portal's Safety Administration. It allows you to check the logged-on status, password status and perform import and export operations.

2.20.1 is_logged_on()

Check if user is logged in to Safety administration offline program

Returns → `bool` Status of login

```
status = safety_admin.is_logged_on()
```

2.20.2 is_password_set()

Check if the safety administration has a password set for the safety offline program

Returns → `bool` Status if password is set

```
status = safety_admin.is_password_set()
```

2.20.3 get_offline_serial_number()

Get the offline serial number

Returns → `str` Offline serial number

```
value = safety_admin.get_offline_serial_number()
```

2.20.4 export_config(target_directory_path: str)

Export the safety administration configuration

The folder “SafetyAdministration” will be automaticcally created on the `target_directory_path`, if not already exists.

Parameters	Type	Description
target_directory_path	str	Folder path for the export

```
safety_admin.export_config(target_directory_path = "C:\\ws\\export")
```

2.20.5 import_config(import_root_directory: str)

Import the safety administration configuration

Parameters	Type	Description
import_root_directory	str	Folder path for the import

```
safety_admin.import_config(import_root_directory =
"C:\\ws\\exported\\SafetyAdministration")
```

2.21 PLC Data - SoftwareUnit

The class represents a software unit in TIA Portal. It provides methods for getting information about the software unit, exporting the software unit and accessing various subcomponents of the software unit, such as PLC tag tables, program blocks, system blocks, user data types and external sources.

2.21.1 get_name()

Get the name of the software unit

Return → `str` Name of the software unit *

```
name = software_unit.get_name()
```

2.21.2 compile()

Compile the software unit

```
software_unit.compile()
```

2.21.3 export_configuration(target_directory_path: str)

Export the software unit configuration

Parameters	Type	Description
target_directory_path	str	Folder path for the export

```
software_unit.export_configuration(target_directory_path = "C:\\ws\\export")
```

2.21.4 get_plc_tag_tables()

Get the list of the PLC tag tables

Returns → `List[PlcTagTable]` List of the PLC tag tables of the software unit

```
pcl_tag_tables = software_unit.get_plc_tag_tables()
```

2.21.5 get_program_blocks()

Get the list of the program blocks

Returns → `List[ProgramBlock]` List of the program blocks of the software unit

```
blocks = software_unit.get_program_blocks()
```

2.21.6 get_system_blocks()

Get the list of the system blocks

Returns → `List[SystemBlock]` List of the system blocks of the software unit

```
blocks = software_unit.get_system_blocks()
```

2.21.7 get_user_data_types()

Get the list of the user data types

Returns → `List[UserDataType]` List of the user data types of the software unit

```
udts = software_unit.get_user_data_types()
```

2.21.8 get_external_sources()

Get the list of the external sources

Returns → `List[ExternalSource]` List of the external sources of the software unit

```
ext_sources = software_unit.get_external_sources()
```

2.21.9 get_named_value_types()

Get the list of the named value types

Returns → `List[NamedValueType]` List of the named value types of the software unit

```
nvts = software_unit.get_named_value_types()
```

2.21.10 export_cross_references(target_directory_path: str, filter: int)

Export cross references of the software unit object filter → integer - [1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

Parameters	Type	Description
target_directory_path	str	Folder path for the export
filter	integer (enum)	[1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

```
software_unit.export_cross_references(target_directory_path =
"C:\\ws\\exportcrossreferences", filter = 2)
```

2.21.11 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.21.12 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.22 PLC Data - SystemBlock

The class represents a system block in TIA Portal. It provides various methods for managing and controlling system blocks.

2.22.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.22.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.22.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.22.4 compile()

Compile the system block

```
systemblock.compile()
```

2.22.5 is_consistent()

Check if the system block is consistent

Returns → `bool` True if consistent

```
value = systemblock.is_consistent()
```

2.22.6 export_cross_references(target_directory_path: str, filter: int)

Export cross references of the system block filter → integer - [1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

Parameters	Type	Description
target_directory_path	str	Folder path for the export
filter	integer (enum)	[1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

```
systemblock.export_cross_references(target_directory_path =
"C:\\ws\\exportcrossreferences", filter = 2)
```

2.22.7 show_in_editor()

Open the PLC object in the editor

```
plc_object.show_in_editor()
```

2.22.8 delete()

Delete the object

```
tiap_object.delete()
```

2.22.9 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.23 PLC Data - TechnologyObject

The class represents a technology object in TIA Portal. It provides various methods for managing and controlling technology objects.

2.23.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.23.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.23.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.23.4 compile()

Compile the technology object

```
to.compile()
```

2.23.5 is_consistent()

Check if the technology object is consistent

Returns → `bool` True if consistent

```
value = to.is_consistent()
```

2.23.6 delete()

Delete the object

```
tiap_object.delete()
```

2.23.7 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.24 PLC Data - UserConstant

The class represents a user constant in TIA Portal. It provides methods for getting information about the user constant, retrieving properties and exporting the constant.

2.24.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.24.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.24.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.24.4 delete()

Delete the object

```
tiap_object.delete()
```

2.24.5 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.25 PLC Data - UserDataType

The class represents a PLC data type in TIA Portal. It provides methods for managing and controlling PLC data types, exporting the data type, retrieving data type information and checking cross-references.

2.25.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.25.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.25.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.25.4 compile()

Compile the PLC data type

```
udt.compile()
```

2.25.5 is_consistent()

Check if the PLC data type is consistent

Returns → `bool` True if consistent

```
value = udt.is_consistent()
```

2.25.6 export_cross_references(target_directory_path: str, filter: int)

Export the cross references of the PLC data type filter → integer - [1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

Parameters	Type	Description
target_directory_path	str	Folder path for the export
filter	integer (enum)	[1: ‘AllObjects’, 2: ‘ObjectsWithReferences’, 3: ‘ObjectsWithoutReferences’, 4: ‘UnusedObjects’]

```
udt.export_cross_references(target_directory_path = "C:\\ws\\exportcrossreferences",
filter = 2)
```

2.25.7 get_type_version_guid()

Get the GUID of the type version

Returns → `str` GUID

```
guid = plc_object.get_type_version_guid()
```

2.25.8 get_type_guid()

Get the GUID of the type

Returns → `str` GUID

```
guid = plc_object.get_type_guid()
```

2.25.9 delete()

Delete the object

```
tiap_object.delete()
```

2.25.10 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.26 PLC Data - WatchTable

The class represents a watch table in TIA Portal. It provides methods for getting information about the watch table, retrieving properties and exporting the table.

2.26.1 get_name()

Get the name of the PLC object

Returns → `str` Name of the PLC object

```
name = plc_object.get_name()
```

2.26.2 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```


2.26.3 export(target_directory_path: str, export_options: Enums.ExportOptions, export_format: Optional[Enums.ExportFormats] = None, keep_folder_structure: Optional[bool] = None)

Export the PLC object

If `keep_folder_structure` is set to True, all group names are represented hierarchically with folders.

Parameters	Type	Description
target_directory_path	str	Folder path for the export
export_options	Enums.ExportOptions	Export options
export_format	Enums.ExportFormats	Export format, by default SimaticML
keep_folder_structure	bool	True: All group names are represented hierarchically with folders

```
plc_object.export(target_directory_path = "C:\\ws\\export", export_options =
Enums.ExportOptions.WithDefaults)
```

2.26.4 is_consistent()

Check if the watch table is consistent

Returns → `bool` True if consistent

```
value = watch_table.is_consistent()
```

2.26.5 show_in_editor()

Open the PLC object in the editor

```
plc_object.show_in_editor()
```

2.26.6 delete()

Delete the object

```
tiap_object.delete()
```

2.26.7 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.27 Project - Project

The class represents a TIA Portal project and offers functions for common project operations, such as saving, closing, enabling/disabling simulation support, archiving, managing hardware and more.

2.27.1 get_portal()

Get the TIA Portal instance from the current project

```
portal = project.get_portal()
```

2.27.2 save()

Save the TIA Portal project

```
project.save()
```

2.27.3 close()

Close the TIA Portal project

```
project.close()
```

2.27.4 set_simulation_support(value: bool)

Set the simulation support for the TIA Portal project

Parameters	Type	Description
value	bool	Set the value of the simulation support

```
project.set_simulation_support(value = True)
```

2.27.5 archive(target_directory_path: str, archive_name: str, delete_existing_archive: bool)

Archive the TIA Portal project

Returns → `str` Full path of the new archive

Parameters	Type	Description
target_directory_path	str	Folder path where archive should be placed
archive_name	str	name of the archive file
delete_existing_archive	bool	Set if the output file should be deleted first

```
archive_fullpath = project.archive(target_directory_path = "C:\\ws\\newfolder",
archive_name = "testproj" , delete_existing_archive = True)
```

2.27.6 save_as(target_directory_path: str, project_name: str)

Save the TIA Portal project under another path and name

Returns → `str` Full path of the saved project

Parameters	Type	Description
target_directory_path	str	Folder path where project should be saved
project_name	str	Name of the project name

```
project_file_path = project.save_as(target_directory_path = "C:\\ws\\newfolder",
project_name = "testproj")
```

2.27.7 export_cax_data(export_file_path: str, log_file_path: str)

Export the CAx data of the TIA Portal project

Returns → `bool` Result state of the export

Parameters	Type	Description
export_file_path	str	Full path of the export file
log_file_path	str	Full path of the log file

```
result = project.export_cax_data(export_file_path =
"C:\\ws\\exportfolder\\exportCAX.aml", log_file_path =
"C:\\ws\\exportfolder\\exportCAX.log")
```

2.27.8 import_cax_data(import_file_path: str, log_file_path: str)

Import CAx data to the TIA Portal project

Returns → `bool` Result state of the import

Parameters	Type	Description
import_file_path	str	Fullpath of the import file
log_file_path	str	Fullpath of the log file

```
result = project.import_cax_data(import_file_path =
"C:\\ws\\importfolder\\importCAX.aml", log_file_path =
"C:\\ws\\importfolder\\importCAX.log")
```

2.27.9 open_topology_editor()

Open the topology editor in TIA Portal for the TIA Portal project

```
project.open_topology_editor()
```

2.27.10 open_network_editor()

Open the network editor in TIA Portal for the TIA Portal project

```
project.open_network_editor()
```

2.27.11 get_plcs()

Get a list of PLCs in the TIA Portal project

Returns → `List[Plc]` All PLC's as list

```
plcs = project.get_plcs()
for plc in plcs:
...
```

2.27.12 get_hmis()

Get a list of HMIs in the TIA Portal project

Returns → `List[Hmi]` All HMI's as list

```
hmis = project.get_hmis()
for hmi in hmis:
...
```

2.27.13 get_application_tests()

Get a list of Test Suite application tests in the TIA Portal project

Needs Test Suite installed and licensed.

Returns → `List[ApplicationTest]` All Application tests as list

```
tests = project.get_application_tests()
```

2.27.14 get_system_tests()

Get a list of Test Suite system tests in the TIA Portal project

Needs Test Suite installed and licensed.

Returns → `List[SystemTest]` All System tests as list

```
tests = project.get_system_tests()
```

2.27.15 get_rule_sets()

Get a list of Test Suite Styleguide rule sets in the TIA Portal project

Needs Test Suite installed and licensed.

Returns → `List[RuleSet]` All Rule sets as list

```
rule_sets = project.get_rule_sets()
```

2.27.16 get_project_library()

Get the project library

Returns → `ProjectLibrary` Project library

```
project_lib = project.get_project_library()
```

2.27.17 web_block_generate()

Generate Web block DBs according the user define pages

Web server must activated.

```
project.web_block_generate()
```

2.27.18 upgrade_hardware(full_upgrade: bool)

Upgrade the devices to the latest order number and firmware version

If `full_upgrade` is set to True, all devices will be changed to the newest available order number (Device type) and firmware:

from CPU1511F 6ES7 511-1FK0**0**-0AB0 **V1.7** to 6ES7 511-1FK0**0**-0AB0 **V1.8**.

With full upgrade: from CPU1511F 6ES7 511-1FK0**0**-0AB0 **V1.7** to 6ES7 511-1FL0**3**-0AB0 **V3.0**.

Parameters	Type	Description
full_upgrade	bool	Set if for full upgrade (latest order number and firmware version)

```
project.upgrade_hardware(full_upgrade = True)
```

2.27.19 sivarc_generate()

Start SiVArc-generation

SiVArc must be installed and licensed.

```
project.sivarc_generate()
```


2.27.20 update_module_description()

Update the module description of all devices in the project

```
project.update_module_description()
```

2.27.21 set_virtual_plc_support(value: bool)

Set virtual PLC support in the project

Available only in V19 or higher

Parameters	Type	Description
value	bool	Set the value of the virtual support

```
project.set_virtual_plc_support(value = "True")
```

2.27.22 import_umac_config(import_file_path: str)

Import UMAC and UMC Configuration to the TIA Portal project

Parameters	Type	Description
import_file_path	str	Full path of the import file
secret	str	Secret for the encryption (optional if passwords encrypted)
secret_env_name	str	Name of the environment variable where Secret is stored (optional if passwords encrypted)

```
project.import_umac_config(import_file_path =
"C:\\ws\\importfolder\\importUMAC.json", secret_env_name = "MYSECRETENV")
```

2.27.23 export_umac_config(export_file_path: str)

Export UMAC and UMC configuration from the TIA Portal project

If export file already exists, it will be overwritten

Parameters	Type	Description
export_file_path	str	Full path of the export file

```
project.export_umac_config(export_file_path =
"C:\\ws\\exportfolder\\exportUMAC.json")
```

2.27.24 encrypt_umac_config(umac_file_path: str, secret: str, secret_env_name: str)

Encrypt UMAC and UMC configuration from the TIA Portal project with provided secret

Parameters	Type	Description
umac_file_path	str	Full path of the umac config file
secret	str	Secret for the encryption
secret_env_name	str	Name of the environment variable where secret value is stored

```
project.encrypt_umac_config(umac_file_path = "C:\\ws\\exportfolder\\exportUMAC.json",
secret = "mySecret")
```

2.27.25 import_password_policy(import_file_path: str)

Import Password Policies to the TIA Portal project

Parameters	Type	Description
import_file_path	str	Full path of the import file

```
project.import_password_policy(import_file_path =
"C:\\ws\\importfolder\\PWPolicy.json")
```

2.27.26 export_password_policy(export_file_path: str)

Export Password Policies from the TIA Portal project

If export file already exists, it will be overwritten

Parameters	Type	Description
export_file_path	str	Full path of the import file

```
project.export_password_policy(export_file_path =
"C:\\ws\\exportfolder\\exportPWPolicy.json")
```

2.27.27 delete()

Delete the TIA Portal project

```
project.delete()
```

2.27.28 start_transaction(undo_text: str, dialog_text: str)

Start exclusive access and new transaction

Parameters	Type	Description
undo_text	str	Text on the Undo button
dialog_text	str	Text on the dialog during the transaction

```
project.start_transaction(undo_text = "MyUndoDescription", dialog_text =
"MyExclusiveAccess")
```

2.27.29 end_transaction(rollback: Optional[bool] = None)

End transaction and exclusive access

Parameters	Type	Description
rollback	bool	Set if changes during transaction should be rolled back (default: false)

```
project.end_transaction()
```


2.27.30 update_transaction(dialog_text: str)

Update transaction of the running exclusive access

Parameters	Type	Description
dialog_text	str	Text on the dialog during the transaction

```
project.update_transaction(dialog_text = "MyNewExclusiveAccess")
```

2.27.31 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.27.32 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.28 Project - ProjectServer

The class represents a TIA Project-Server in TIA Portal.

2.28.1 get_host()

Get the host of the TIA Project-Server
Returns → `str` Host

```
host = server_project.get_host()
```

2.28.2 get_port()

Get the port of the TIA Project-Server
Returns → `int` Port number

```
port = server_project.get_port()
```

2.28.3 get_server_name()

Get the server name of the TIA Project-Server
Returns → `str` Server name

```
name = server_project.get_server_name()
```

2.28.4 print_info()

Show information of the TIA Project-Server

```
server_project.print_info()
```

2.28.5 get_property(name: str)

Get the property of the object
Properties which are not string will be converted to string if possible.
Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.28.6 get_properties()

Get all properties of the object
Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

2.29 Test Suite - ApplicationTest

The class represents an Test Suite application test in the context of TIA Portal.
Test Suite Advanced must be installed and licensed.

2.29.1 get_name()

Get the name of the Test Suite application test

Return → `str` Name of the application test

```
name = app_test.get_name()
```

2.29.2 get_property(name: str)

Get the property of the Test Suite application test

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property of the Application Test

```
property_value = app_test.get_property(name = "Property")
```

2.29.3 export(target_directory_path: str)

Export the Test Suite application test

The folder “Application tests” will be automatically created on the `target_directory_path`, if not already exists.

Parameters	Type	Description
target_directory_path	str	Folder path for the export

```
app_test.export(target_directory_path = "C:\\ws\\export")
```

2.29.4 set_scope(plc_name: str, instance_name: Optional[str] = None, execution_mode: Optional[int] = None)

Set the scope of the Test Suite application test

Parameters	Type	Description
plc_name	str	Name of the PLC to be used
instance_name	str	PLCSIM Instance name (only V19 and higher)
execution_mode	integer (enum)	(only V19 and higher) [1: ‘SystemManagedPLCSIMInstance’, 2: ‘ExternallyManagedPLCSIMInstance’]

```
app_test.set_scope(plc_name = "PLC_1")
```

2.30 Test Suite - RuleSet

The class represents a Test Suite style guide rule set in the context of TIA Portal.
Test Suite Advanced must be installed and licensed.

2.30.1 get_name()

Get the name of the Test Suite style guide rule set
Return → `str` Name of the rule set

```
name = rule.get_name()
```

2.30.2 get_property(name: str)

Get the property of the Test Suite style guide rule set
Properties which are not string will be converted to string if possible.
Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property of the rule set

```
property_value = rule.get_property(name = "Property")
```

2.30.3 export(target_directory_path: str)

Export the Test Suite style guide rule set
The folder “Style guide” will be automatically created on the `target_directory_path`, if not already exists.

Parameters	Type	Description
target_directory_path	str	Folder path for the export

```
rule.export(target_directory_path = "C:\\ws\\export")
```

2.31 Test Suite - SystemTest

The class represents a Test Suite system test in the context of TIA Portal.
Test Suite Advanced must be installed and licensed.

2.31.1 get_name()

Get the name of the Test Suite system test
Returns → `str` Name of the system test

```
name = sys_test.get_name()
```

2.31.2 export(target_directory_path: str)

Export the Test Suite system test
The folder “System tests” will be automatically created on the `target_directory_path`, if not already exists.

Parameters	Type	Description
target_directory_path	str	Folder path for the export

```
sys_test.export(target_directory_path = "C:\\ws\\export")
```

2.31.3 set_scope(opcua_server_address: str, opcua_server_interface_type: int, opcua_server_interface_folder_path: Optional[str] = None)

Set the scope of the Test Suite system test

Parameters	Type	Description
opcua_server_address	str	OPC UA server adress e.g. opc.tcp://server.port/path
opcua_server_interface_type	integer (enum)	[1: 'UserDefined', 2: 'StandardSIMATIC']
opcua_server_interface_folder_path	str	Folder path to interface files

```
sys_test.set_scope(opcua_server_address = "opc.tcp://server.port/path",
opcua_server_interface_type = 1 )
```

2.31.4 get_property(name: str)

Get the property of the object

Properties which are not string will be converted to string if possible.

Returns → `str` Value of the property as string

Parameters	Type	Description
name	str	Property name of the object

```
property_value = tiap_object.get_property(name = "CreationDate")
```

2.31.5 get_properties()

Get all properties of the object

Returns → `List[str]` Names of the attributes

```
properties = tiap_object.get_properties()
```

3 Appendix

3.1 Service and support

SiePortal

The integrated platform for product selection, purchasing and support - and connection of Industry Mall and Online support. The SiePortal home page replaces the previous home pages of the Industry Mall and the Online Support Portal (SIOS) and combines them.

- Products & Services
  In Products & Services, you can find all our offerings as previously available in Mall Catalog.
- Support
  In Support, you can find all information helpful for resolving technical issues with our products.
- mySieportal
  mySiePortal collects all your personal data and processes, from your account to current orders, service requests and more. You can only see the full range of functions here after you have logged in.

You can access SiePortal via this address: sieportal.siemens.com

Industry Online Support

Industry Online Support is the previous address for information on our products, solutions and services.

Product information, manuals, downloads, FAQs and application examples - all information is available with just a few mouse clicks: support.industry.siemens.com

Technical Support

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

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.
For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page: siemens.com/sitrain

Industry Online Support app

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

3.2 Application support

Siemens AG
Digital Factory Division
Factory Automation

3.3 Links and literature

Table 4-1

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/109742322

3.4 Change documentation

Table 4-2

Version	Date	Modifications
V1.0.7	11/2024	First release-version