SIEMENS

TIA Portal Test Suite

Function Manual

Valid for TIA Portal Test Suite

01/2023
A5E50009374-AA

Legal information

Warning notice system

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

⚠ WARNING

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

Trademarks

All names identified by ® are registered trademarks of Siemens 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.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E50009374-AA
Ⓟ 10/2023 Subject to change

Table of contents

Style guide

1

1.1 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.

1.2 Introduction

Overview

This feature enables you to define a rule set which contains several rules against them, the following PLC objects are validated:

- Function Blocks
- Functions
- Organization Blocks
- Varities of global DB/Instance DB
- PLC tags
- User Defined datatypes (UDT)

Benefits

- Easy definition of rules within TIA Portal
- Fast tracking and solving of violations, as the user can directly go to violation location
- Good program quality is guaranteed

1.3 Creating and Managing rule sets

Introduction

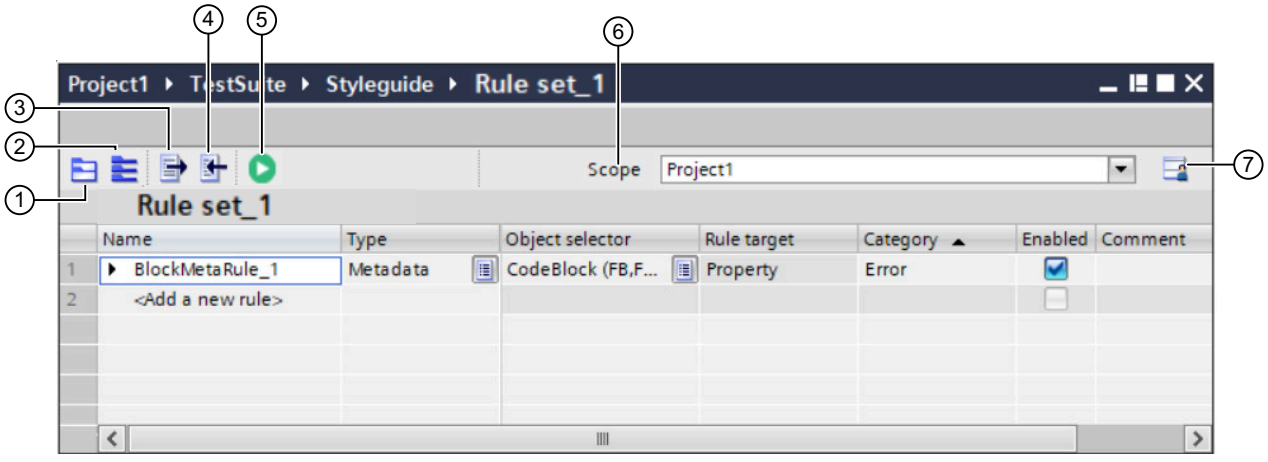
This chapter guides you to create and manage rule sets. It also assists you in performing general rule set operations.

Rule set editor

The rule set editor allows you to edit each rule set individually. It also enables you to peform the rule set operations individually for each rule set.

Structure of a rule set

Description 1



(1) Expands all the rows of the dialog

(2) Collapses all the rows of the dialog

(3) Exports all the rules

(4) Imports all the rules

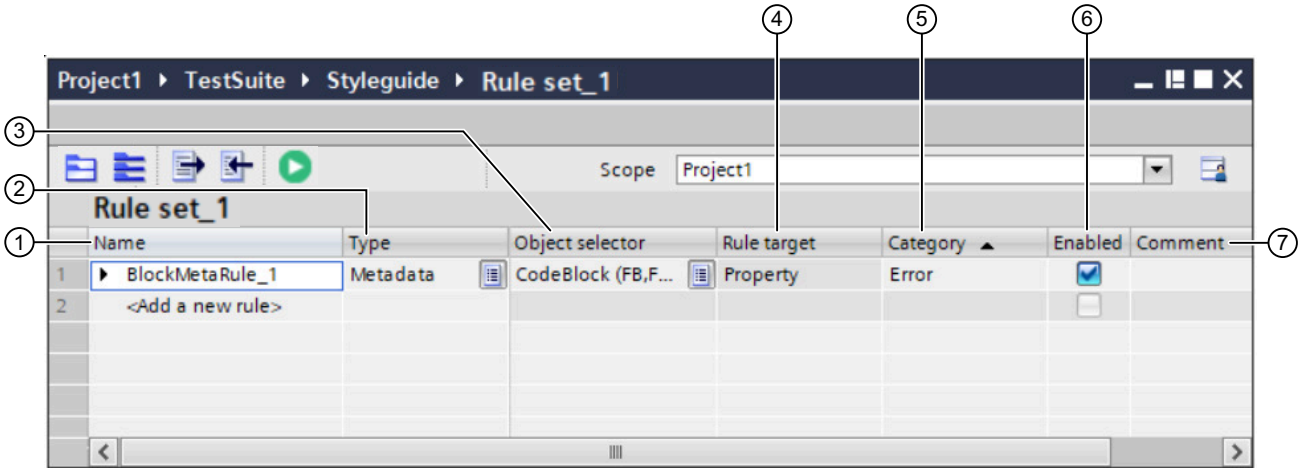
(5) Run the rule set

(6) The scope drop-down list contains the following properties:

- The inclusion blocks for rule set execution is displayed on the editor Toolbar. It is displayed using a checked tree view.
- The look and feel for the checked tree view is same as the global search.
- The PLC and the blocks in which the rules will run are selected from the Scope selector window.

(7) Saves the current window settings

Description 2



(1) Describes the name of the rule You can also edit this according to your requirements.

(2) Allows you to select the type of rule.

(3) Allows you to select the object type for the specified Rule.

(4) Describes the rule target of the specified Rule.

(5) Describes the category of the specified Rule.

(6) Allows you to enable the specified rule by selecting the check box.

(7) Allows you to add comments for the specified rule.

Prerequisites

- STEP 7 Professional V19 installed.
- Optional Package TIA Portal Test Suite Advanced V19 is installed.

Note

As an example, we will use "Rule set_1" throughout the style guide chapter. The user can use any Rule set name when creating a style guide.

Adding a new rule set

You can define a rule set which contains several rules against the PLC objects that can be validated. The rule sets are added as explained below:

Procedure

1. In the Test Suite project that you have created, double click "Add new Rule Set". A new rule set "Rule set_1" is added to the project tree.

Note

- Rule sets are sorted automatically after adding a new entry to the project tree.
- Rule sets are sorted as per the precedence of unicode characters (Similar to project tree objects sorting).

Deleting a rule set

You can delete a rule set from the project tree and deleting will remove the rule set from the project tree under the "Style guide: subfolder.

Procedure 1

1. Under the "Project tree" area, select "Project 1 > Test Suite > Style guide > Rule set_1".
2. Right click "Rule set_1 > Delete".
3. The selected rule set is now deleted.

Note

"Confirm deleting" dialog box will appear after you select rule set(s) to be deleted. You can choose "Yes" or "No" based on your requirements.

Renaming a rule set

You can rename a rule set in the project tree. After you rename a rule set(s), they will be sorted in an order of character precedence similar to the program blocks folder. You can rename a rule set(s) using the following methods:

Procedure

1. Under the "Project tree" area, select "Project 1 > Test Suite > Style guide > Rule set_1".
2. Right click "Rule set_1 > Rename".
3. Rename the rule set accroding to your choice.

Note

- A rule set name can accept a maximum of 125 unicode charaters
- When multiple rule sets are selected, you can rename the last selected rule set.
- All unicode characters are accepted except the followinf windows characters that are reserved
  - < (less than)
  - > (greater than)
  - : (colon)
  - " (double quote)
  - / (forward slash)
  - \ (backslash)
  - | (vertical bar or pipe)
  - ? (question mark)
  - * (asterisk)
  - white space

Opening a rule set editor

You can open a rule set editor and edit it separetely by the following methods:

1. Double click on the required rule set.
2. Select the required rule set and click "Open"
3. Select the required rule set and press "enter"
4. Select the required rule set, pause and use mouse click.

Note

You also have the option to open multiple rule sets.

Copy/Paste and Drag/drop a rule set from project tree

You can copy rule set(s) from the project tree and copying them will add the rule set(s) to the respective project tree under the "Style guide" subfolder with the name similar to project tree objects. It can be done in the followiing ways"

1. Press "Ctrl + C".
2. Click "Copy" from the context menu.
3. Click "Copy" from the main menu.
4. Tool bar icon

Note

- You can copy single/multiple rule sets from the project tree.
- Rule sets are sorted automatically after you perform the paste operation based on the precedence of the unicode char

1.4 Rule Types

1.4.1 Casing

Introduction

This feature allows you to perform a capitalization check for a particular type of PLC objects selected. This style guide check will suport rules to check how names of certain PLC objects are written. For example: If a name is written in camel case or pascal case, or in upper case.

Prerequisites

- TIA Portal V19 is open.
- Test Suite project is opened in the project view
- You have created a rule set

Properties

The following properties are defined for this rule type:

- Casing:
  - Upper casing: All appearing characters are in capital letters and the first character should always be an alphabetic letter.
  - Camel casing: First character is always in lower case and should always be an alphabetic letter. In front of an uppercase character, there must be atleast one lower case character. If a character follows a capital letter, then it must be in lower case. If a character follows a non alphabetic letter, then it must be in lower case.
  - Pascal casing: First alphabetic character must always be in upper case and should always be an alphabetic letter. If a character follows a capital letter, then it must be in lower case. If a character follows a non alphabetic letter, then it must be in upper case.
- Violation condition:
  - Is set: the casing property is set
  - Is not set: the casing property is not set

Procedure

To run the casing check rule for "Rule set_1":

1. Under the "Project tree" area, select "Project 1 > Test Suite > Style guide".
2. Double click "Rule set_1". The style guide editor appears. In the "Type" column, Click 
3. Select the "Casing" rule from the drop-down list.

   Note

   Once you select the required "Rule set" rule in the name column, the name of the rule set appears in the "Name" column automatically. You can modify the name of the rule set according to your choice.

4. Once you select the "Casingrule_1" rule, the following properties of the rule gets added to the "Name" column
   - Casing: Upper casing, Camel casing, and Pascal casing
   - Violation condition: Is set and Is not set

   Select the "Violation condition" as "Upper casing" or "Camel casing" or "Pascal casing" according to your requirements.
5. Select "Is set" or "Is not set" as the "Violation condition" depending on your requirement.
6. To run the style guide rule, click 

Example

Casing Type	Violation Condition	Object Name	Test Result
Upper Casing	Is Set	“CASING” “CASING123” "123CASING" “$&%CASING” “Casing1” “CaSiNd1234” “casing1234CASING” “123” “U” “UPPER_CASING” “upper”	Should Identify the objects "CASING" "CASING123" "U" "UPPER_CASING"
Upper Casing	Is Not Set	"CASING" "CASING123" "123CASING" "$&%CASING" "Casing1" "CaSiNd1234" "casing1234CASING" "123" "U" "UPPER_CASING" "upper"	Should Identify the objects "123CASING" "$&%CASING" "Casing1" "CaSiNd1234" "casing1234CASING" "123" "upper"
Pascal Casing	Is Set	"Casing" "CASING123" "$&%Casing" "@Casing" "Casing1Casing2" "Casing1casing2" "CAsing1CAsing2" "P" "P1" "PascalC" "PascalC1" "PascalC1Test" "123" "PA" "Pascal_Casing"	Should Identify the objects “Casing” “Casing1Casing2” “P” “P1” “PascalC” “PascalC1” “PascalC1Test” “Pascal_Casing”

Casing Type	Violation Condition	Object Name	Test Result
Pascal Casing	Is Not Set	"Casing" "CASING123" "$&%Casing" "@Casing" "Casing1Casing2" "Casing1casing2" "CAsing1CAsing2" "P" "P1" "PascalC" "PascalC1" "PascalC1Test" "123" "PA" "Pascal_Casing"	Should identify the objects “CASING123” “Casing1casing2” “CAsing1CAsing2” “$&%Casing” “@Casing” “123”

Casing Type	Violation Condition	Object Name	Test Result
Camel Casing	Is Set	"caSing" "caSing123" "$&%cAsing" "@casing" "casing" "casinG1casing2" "casing1Casing2Casing3" "cASING" "Casing" "camel1c" "c" "123" "camel_Casing" "camelC" "camelC1"	Should identify the objects “caSing” “caSing123” “casing” “casinG1casing2” “camel1c” “c” “camelC” “camelC1”
Camel Casing	Is Not Set	"caSing" "caSing123" "$&%cAsing" "@casing" "casing" "casinG1casing2" "casing1Casing2Casing3" "cASING" "Casing" "camel1c" "c" "123" "camel_Casing" "camelC" "camelC1"	Should identify the objects “casing1Casing2Casing3” “cASING” “Casing” “camel_Casing“ "$&cAsing” “@casing” “123“

1.4.2 Name Contains

Introduction

This feature allows you to perform a rule set check that will support rules to check if names of certain objects of a PLC device contains or not contains the given characters.

Prerequisites

- TIA Portal V19 is open
- Test Suite project is opened in the project view
- A rule set is created

Properties

The following properties are defined for this rule type:

- Violation condition Contains, Does not contains.

Procedure

To run the Name contains rule for "Style guide_1":

1. Under the "Project tree" area, select "Project 1 > Test Suite > Style guide".
2. Double click "Rule set_1". The rule set editor appears. In the "Type" column, Click 

   Note

   Once you select the required rule set in the name column, the name of the rule set appears in the "Name" column automatically. You can modify the name of the rule set according to your choice.

3. Select the "Name contains" rule from the drop-down list. The "ContainsRule_1' appears in the "Name" column automatically.
4. Once you select the "Name contains" rule, the following properties of the rule gets added to the "Name" column:
   - Violation Condition: "Contains" or "Not contains"
   - Value: One or more characters within the range 1-128
   - Match type: "Case sensitive" or "Case insensitive"

   Select the "Violation condition" as "Contains" or "Not contains" according to your requirements
5. Insert any "Value"
6. To run the rule set check. click 

Example

Violation Condition	Match type	String Specified	Object Name	Test Result
Contains	Case insensitive	"check"	“To_Check” “To_Ch_ec_k” “To_check	Identifies the objects “To_Check” “To_check”
	Case sensitive	"Check"	“To_Check” “To_Ch_ec_k” “To_check	Identifies the objects “To_Check”
Not contains	Case insensitive	"check"	“To_Check” “To_Ch_ec_k” “To_check	Identifies the objects “To_Ch_ec_k”
	Case sensitive	"Check"	“To_Check” “To_Ch_ec_k” “To_check"	Identifies the objects “To_Ch_ec_k” “To_check"
Contains	Case insensitive	“ name”	“ name123” “name123” “name 123” “_ name123”	Identifies the objects “ name123” “_ name123”
	Case sensitive	"Name"	“ name123” “Name123” “name 123” “_ name123”	Identifies the objects “Name123”
Not contains	Case insensitive	“ name”	“ name123” “Name123” “name 123” “_ name123”	Identifies the objects “Name123” “name 123”
	Case sensitive	"Name"	“ name123” “Name123” “name 123” “_ name123”	Identifies the objects “name 123” “_ name123” “ name123”

1.4.3 Metadata

Introduction

This feature allows you to perform a metadata check on the style guide. The style guide check will support rules to check the metadata of certain objects of a PLC device.

Prerequisites

- TIA Portal V19 is open
- Test Suite project is opened in the project view
- A Rule set is created

Properties

The following rule attributes are defined for this rule type:

Category 1

- Violation Condition: Contains, Exists, Not Contains, Not exists
- Property (only for the above violation conditions): Author, Comment, Family, Title, User-defined ID, Version
- Value: any string value

Category 2

- Violation Condition: Is set, Is not set
- Property: (Only for the above violation conditions): Automatic numbering, Enable tag readback, Handle Errors within block, IEC check, Manual numbering, Multiple instance capability, Optimized block access, Set ENO automatically.

Extend "Comment" property for additional targets

The following object selector list is used for validating the "Comment" property:

- CodeBlock.Interface (In,Out,InOut)
- CodeBlock.Interface.Input
- CodeBlock.Interface.Output
- CodeBlock.Interface.InOut
- CodeBlock.Interface.Static
- CodeBlock.Interface.Temp
- CodeBlock.Interface.Constant
- GlobalDataBlock.Static
- PLCDataType
- PLCDataType.Member
- PLCTagTable
- PLCTagTable.Constants

Note

This rule considers to validate the comments in the current editing language.

Procedure

To run the meta data rule for "Rule set_1":

1. Under the "Project tree" area, select "Project 1 > Test Suite > Style guide".
2. Double click "Rule set_1". The style guide editor appears. In the "Type" column, Click
3. Select the "Metadata" rule from the drop-down list.
4. Once you select the "MetaRule_1", the properties gets displayed as explained in the "Properties" section of this chapter.
5. If you select the "violation condition" as "Contains" or "Exists" or "Not contains" or "Not exists", the following "Property" is available for selection. As an example, we have selected the "Contains" as the violation condition here.
6. Enter any string value.
7. To run the style guide rule. Click
8. If you select the "Violation condition" as "Is set" or "Is not set", the following "Property" options are available. As an example, we have selected the "Violation condition" as "Is set":
9. There is no "value" to be entered here. Now, enter any string value
10. To run the style guide rule for the above properties selected, click

Note

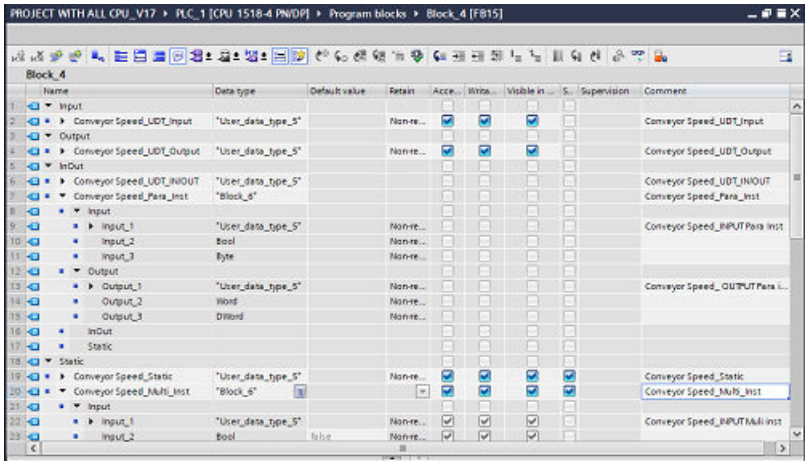
The elements of User defined data type (UDT) instances in blocks, Parameter instance and Multi instance are not validated.

For an array of UDT, only the parent is validated.

For DB of UDT, no elements are validated.

For an Array DB, only the parent is validated.

For Example, in the following image, the elements of UDT (Conveyor Speed_UDT), Parameter instance (Conveyor Speed_Para_Inst) and Multi instance (Conveyor Speed_Multi_Inst) will not be validated, but the array element will be validated.



1.4.4 Name Length

Introduction

This rule checks the length of names for specific objects of a PLC device.

Prerequisites

- TIA Portal V19 is open
- Test Suite project is opened in the project view
- A rule set is created

Properties

The following properties are defined for this rule type:

- Less than: Identifies the objects which has the characters in the name less than the specified value.
- Greater than: Identifies the objects which has the characters in the name greater than the specified value
- Value: Numerical value within the range 2-127

Procedure

To run the name length check rule for "Rule set_1":

1. Under the "Project tree" area, select "Project 1 > Test Suite > Style guide".
2. Double click "Rule set_1". The rule set editor appears.

   Note

   Once you select the required rule set in the name column, the name of the rule set appears in the "Name" column automatically. You can modify the name of the rule set according to your choice.

3. In the "Type" column, Click 
4. Select the "Name length" rule from the drop-down list. Once you select the "Name length" rule, the following properties of the rule gets added to the "Name" column:
   - Violation Condition: "Greater than" or "less than"
   - Value: Numerical value within the range 2-127

   Select the "Violation condition" as "Greater than" or "Less than" according to your requirements:
5. Enter any numerical value as shown.
6. To run the rule set check, click 

Example

Violation Condition	Value Specified in the project tree	Test Result
Less than	2	Identifies the objects that are named with single character. A name with less than one is not possible
Greater than	127	Identifies the objects that are named with 128 characters. Names with more than 128 characters are not allowed in TIA.
Greater than	2	Identifies objects named with a length of 3 - 128 characters.
Less than	10	Identifies the objects, named with a length 1 to 9 characters.

1.4.5 Prefix Suffix

Introduction

This rule set check allows you to support rules to check if the names of certain objects of a PLC string start or end with a given string.

Prerequisites

- TIA Portal V19 is open
- Test Suite project is opened in the project view
- A rule set is created

Properties

The following properties are defined for this rule type:

- Context: Starts with or Ends with
- Expected String: Prefix or Suffix that must be part of this name
- Violation condition: Is equal, Is not equal
- Value: Supported range is 1 - 128

Procedure

To run the "Prefix suffix" rule for "Rule set_1":

1. Under the "Project tree" area, select "Project 1 > Test Suite > Style guide".
2. Double click "Rule set_1". The style guide editor appears. In the "Type" column, Click 
3. Select the "Prefix suffix" rule from the drop-down list. The "PrefixSuffixRule_1' appears in the "Name" column automatically

   Note

   Once you select the required rule set in the name column, the name of the rule set appears in the "Name" column automatically. You can modify the name of the rule set according to your choice.

4. Select the "Starts with" or "Ends with" according to your requirements:
5. Select the "Violation condition" as "Is equal" or "Is not equal" according to your requirements
6. Insert any numerical value
7. Select Match type as "Case insensitive" or "Case sensitive".
8. To run the rule set check, click 

Example

Violation Condition	Match type	String Specified	Object Name	Test Result
Starts with	Case insensitive	“Pre”	“precondition” “Precondition” " Precondition" “_Precondition” “@Precondition”	Identifies the objects “precondition” “Precondition”
	Case sensitive	“Pre”	“precondition” “Precondition” " Precondition" “_Precondition” “@Precondition”	Identifies the objects “Precondition”

Violation Condition	Match type	String Specified	Object Name	Test Result
Starts with	Case insensitive	" Pre"	" precondition" "Precondition" " Precondition" "_Precondition" "@Precondition" "name Precondition"	Identifies the objects " Precondition" " precondition"
	Case sensitive	" Pre"	" precondition" "Precondition" " Precondition" "_Precondition" "@Precondition" "name Precondition"	Identifies the objects " Precondition"
Ends with	Case insensitive	"condition"	"preconditional" "Precondition" " Preconditional" "_Precondition" "@Precondition"	Identifies the objects "Precondition" "_Precondition" "@Precondition"
	Case sensitive	"Condition"	"preconditional" "PreCondition" "Preconditional" "_Precondition" "@Precondition"	Identifies the objects "PreCondition"
Ends with	Case insensitive	"tioN"	"preconditional" "Precondition" "PreconditioN" " Preconditional" "_Precondition_" "@Precondition"	Identifies the objects "Precondition" "PreconditioN" "@Precondition"
	Case sensitive	"tioN"	"preconditional" "Precondition" "PreconditioN" " Preconditional" "_Precondition_" "@Precondition"	Identifies the objects "PreconditioN"

1.5 Object Selector

Introduction

The object selector provides you the list of objects which are applicable for the rule type selected. The information is as shown below:

Rule Type	Objects Listed
Name length / Name contains /Prefix Suffix/Casing	CodeBlock (FB,FC,OB) CodeBlock.Function CodeBlock.FunctionBlock CodeBlock.Interface (In,Out,InOut) CodeBlock.Interface.Input CodeBlock.Interface.Output CodeBlock.Interface.InOut CodeBlock.Interface.Static CodeBlock.Interface.Temp CodeBlock.Interface.Constant CodeBlock.OrganisationBlock GlobalDataBlock GlobalDataBlock.Static InstanceDataBlock PLCDataType PLCDataType.Member PLCTagTable PLCTagTable.Tags PLCTagTable.Constants
Metadata	CodeBlock (FB,FC,OB) CodeBlock.Function CodeBlock.FunctionBlock CodeBlock.Interface (In,Out,InOut) CodeBlock.Interface.Input CodeBlock.Interface.Output CodeBlock.Interface.InOut CodeBlock.Interface.Static CodeBlock.OrganisationBlock GlobalDataBlock GlobalDataBlock.Static InstanceDataBlock PLCDataType PLCDataType.Member PLCTagTable.Tags

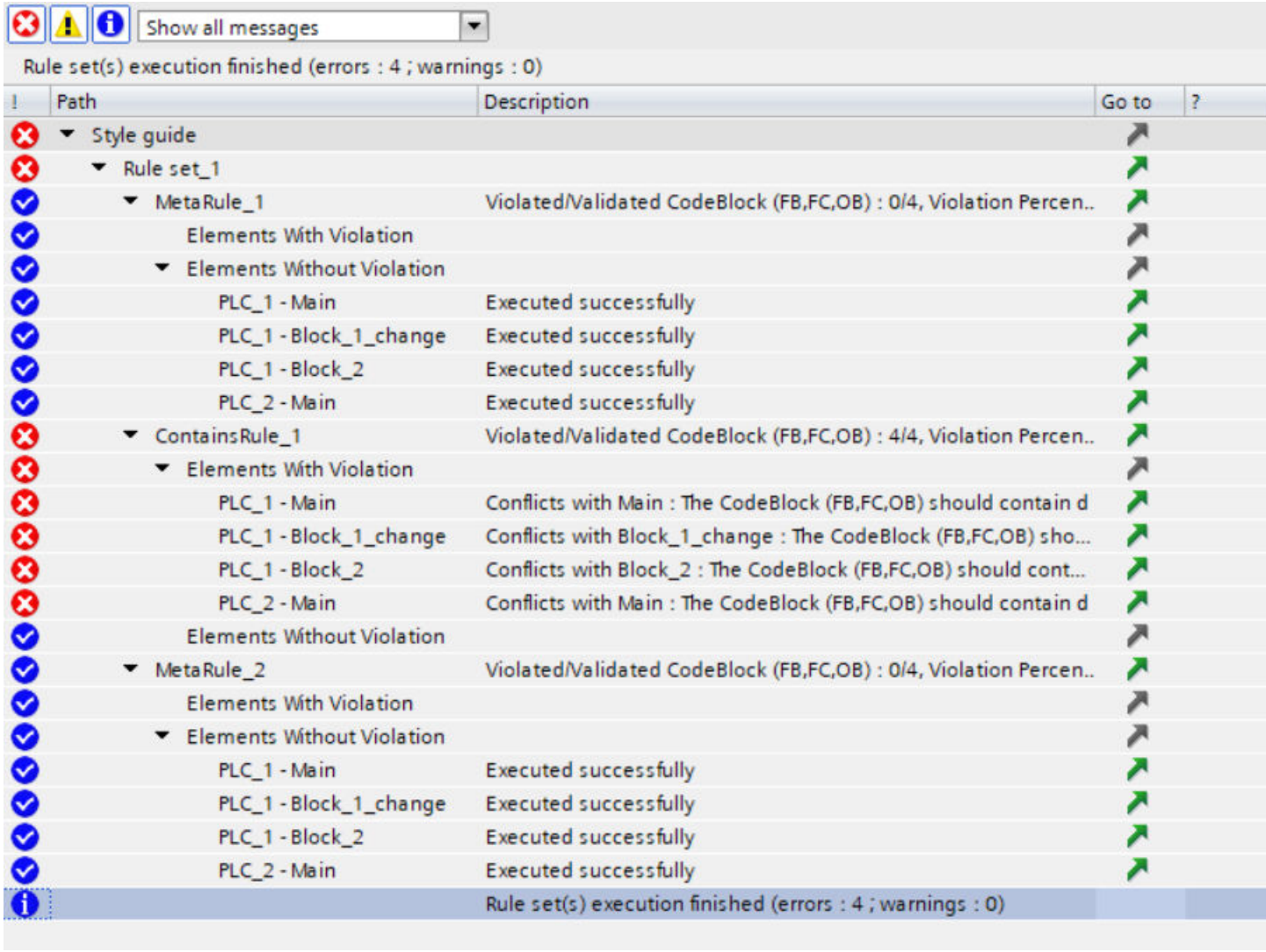
Note

Separate object selectors for organization block, function and function block are available from V18 onwards.

1.6 Display of test results

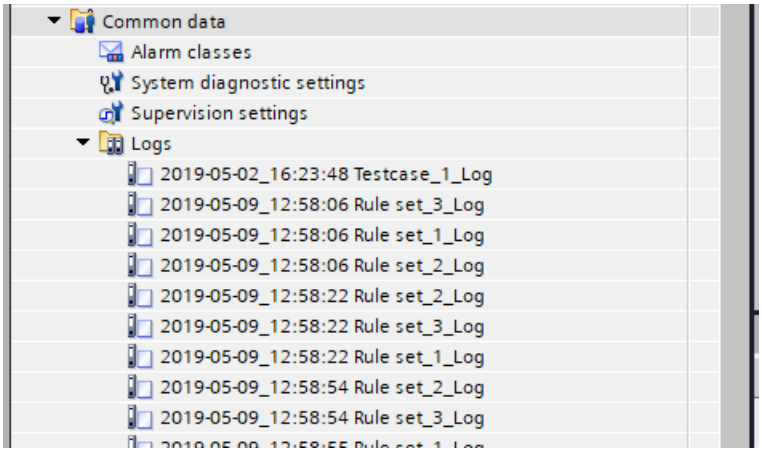
Test Result display and Log File Creation for Rule set Execution

Execution of a rule set will generate violations, and these messages are displayed in the "Test Results" tab page in the inspector frame. Additionaly, this report is saved in a log file that is stored in the folder "Common Data\Logs" in the TIA project. This report is displayed in the common log viewer of the TIA Portal. Each logged violation provides a "Go to" to the location where the violation occurred. Each rule set will have a separate log file generated. It contains the Test results of the rules defined inside the rule set. This is the same behavior for the rule sets defined inside the groups and sub-groups.



Log Results

Execution of the rule set(s) will generate the violations occurred on the selected target objects. It is saved in a log file that is stored in the folder "Common Data\Logs" in project tree and the report is displayed in the common log viewer of the TIA Portal.



Note

The below mentioned objects are excluded from the execution:

- System generated objects under system blocks folder.
- Read only objects.

1.7 Additional Features

Introduction

Style guide checker also supports the following features:

1. Master copy support
2. Import/Export
3. Multiuser engineering

Master Copy Support

Mater copy support is used to transmit rule sets defined in style guide checker to other projects. It is possible to create a master copy out of a rule set. Master copies are also used to create standardized copies of frequently used rule sets.You can create as many items as needed and then insert them into the project.

You can store master copies either in the project library or in a global library. Master copies in the project library can only be used within the project. When you create a master copy in a global library, it can be used in different projects.

Limitations of Master Copy Support

- Drag and drop of Test Suite library objects into the project tree is not supported. Therefore, use standard copy and paste instead.
- Copy and paste of user folders from library to Test Suite project tree below is not supported.

Import/Export

Rules from rule set editor

You can export and import the rules added in the rule set editor to an external file with the context menu/tool bar items provided in the editor. Rules which are defined inside the rule set editor can be exported to an XML external source file and the rules can also be imported to the rule set editor from an external XML source file.

Rule sets in project tree

You can import and export the rule set(s) in project tree to an external source file with the following options:

- **Import**: Context menu entry "Import rule set(s)" in project tree when you select Style guide folder.
- **Export**: Context menu entry "Export rule set(s)" in project tree when you select Style guide folder or rule set(s).

Rule set(s) can be imported/exported to an external source file with .XML extension.

Errors and Warnings messages for import operation:

You will also be shown status information, warnings and error messages in the Inspector window under the "Info > General" tab for import operation.

Below are the different feedback messages provided for import operation:

- Import of rule set(s) completed with warnings, when:
  - The rule set property 'Author' imported with the value which is not valid (Example: more than 125 characters/only blank space).
  - The rule set property 'Version' imported with the value which is not valid (Example: outside the range 0.0 to 99.99).
  - Some of the rules in a rule set are failed to import because of wrong configuration.
    Example for wrong rule configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--exported by 'TIA Portal Test Suite Advanced' Version 17.00.00.00 © 2004-2016 Siemens AG - all rights reserved-->
- <RuleSets>
  - <RuleSet Name="Rule set_1">
    - <Property>
        <PropertyAttribute Name="Author" Value=""/>
        <PropertyAttribute Name="Version" Value="0.1"/>
        <PropertyAttribute Name="Comment" Value=""/>
      </Property>
    - <Rules>
      - <Rule Name="MetaRule_1" Enabled="True" Type="MetaDataRule">
          <RuleAttribute Name="ObjectSelector_WrongConfiguration" Value="CodeBlock_FB_FC_OB"/>
          <RuleAttribute Name="Target" Value="TargetProperty"/>
          <RuleAttribute Name="Category" Value="Error"/>
          <RuleAttribute Name="Comment" Value=""/>
          <RuleAttribute Name="Property" Value="Comment"/>
          <RuleAttribute Name="ViolationCondition" Value="Contains"/>
          <RuleAttribute Name="Value" Value="ABC"/>
        </Rule>
      </Rules>
    </RuleSet>
</RuleSets>
```

  - Some of the rules in a rule set are imported with default configuration because of missed attributes.
    Example for missed attributes of a rule:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--exported by 'TIA Portal Test Suite Advanced' Version 17.00.00.00 © 2004-2016 Siemens AG - all rights reserved-->
- <RuleSets>
  - <RuleSet Name="Rule set_1">
    - <Property>
        <PropertyAttribute Name="Author" Value=""/>
        <PropertyAttribute Name="Version" Value="0.1"/>
        <PropertyAttribute Name="Comment" Value=""/>
      </Property>
    - <Rules>
      - <Rule Name="MetaRule_2" Enabled="True" Type="MetaDataRule">
          <!--Missed Attribute : ObjectSelector <RuleAttribute Name='ObjectSelector' Value='CodeBlock_FB_FC_OB' />-->
          <RuleAttribute Name="Target" Value="TargetProperty"/>
          <RuleAttribute Name="Category" Value="Error"/>
          <RuleAttribute Name="Comment" Value=""/>
          <RuleAttribute Name="Property" Value="Comment"/>
          <RuleAttribute Name="ViolationCondition" Value="Contains"/>
          <RuleAttribute Name="Value" Value="ABC"/>
        </Rule>
      </Rules>
    </RuleSet>
</RuleSets>
```

- Import of rule set(s) completed with errors, when:
  - Rule set name is missing.
  - Rule set name contains the characters which are not allowed in the TIA project (Example: <, >, :, ", /, \, |, ?, *, space).
  - Rule set name contains only NULL/empty string.

- <RuleSet> and </RuleSet> tag missing.
- <Rules> and </Rules> tag missing.
- <RuleSet> /</RuleSet> mentioned inside <Rules> and </Rules>.
- <RuleSets> /</RuleSets> mentioned inside <Rules> and </Rules>.
- <RuleSets> /</RuleSets> mentioned inside <RuleSet> and </RuleSet>.
- <Rules> /</Rules> not enclosed within <RuleSet> /</RuleSet>.
- <RuleSet> /</RuleSet> not enclosed within <RuleSets> /</RuleSets>.

Note

<Property> /</Property> is not mandatory tag in the file to import and it will be considered to import only if it is available in the very beginning of tag <RuleSet>/</RuleSet>.

When you abort the import operation then the rule set which is already in progress to import will be terminated immediately.

Multiuser engineering

Style guide supports modification and execution of rule sets in Multiuser engineering environment.

Note

- Rule set will be marked "Red" when you copy/paste rule set(s) across different local sessions.
- Rule set execution marks the object whenever there is a change in the selected rule set scope.
- "Check -in" or "Refresh" action will set the rule set scope in the target view to default (Project selection) when the selected scope is not available in the target view. The target view can be Project server / Local session.

1.8 Additional Property

Introduction

The Additional Property feature provides the user with the option to add a condition to a specific rule, so that the rule is validated only when the condition is satisfied with the specified additional property.

The condition can be set with the help of the following properties:

- Member type
- Data type
- Block type

Member type

The member type property is provided to validate the rule for the below listed member types of the variables defined in the program blocks.

- Array
- Struct
- Block instance
- Array of Block instances

The target object selectors for which the above property will be applicable are as follows:

- CodeBlock.Interface (In,Out,InOut)
- CodeBlock.Interface.Input
- CodeBlock.Interface.Output
- CodeBlock.Interface.InOut
- CodeBlock.Interface.Static
- CodeBlock.Interface.Temp
- GlobalDataBlock.Static
- PLCDataType.Member

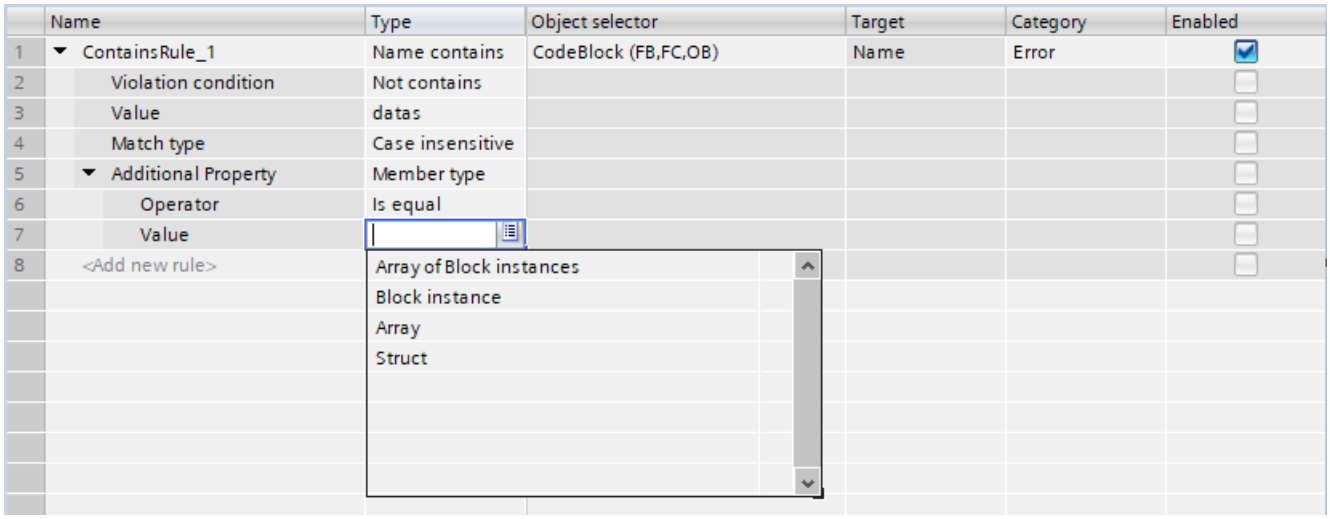
Note

Rule execution for other object selectors will be ignored.

Examples

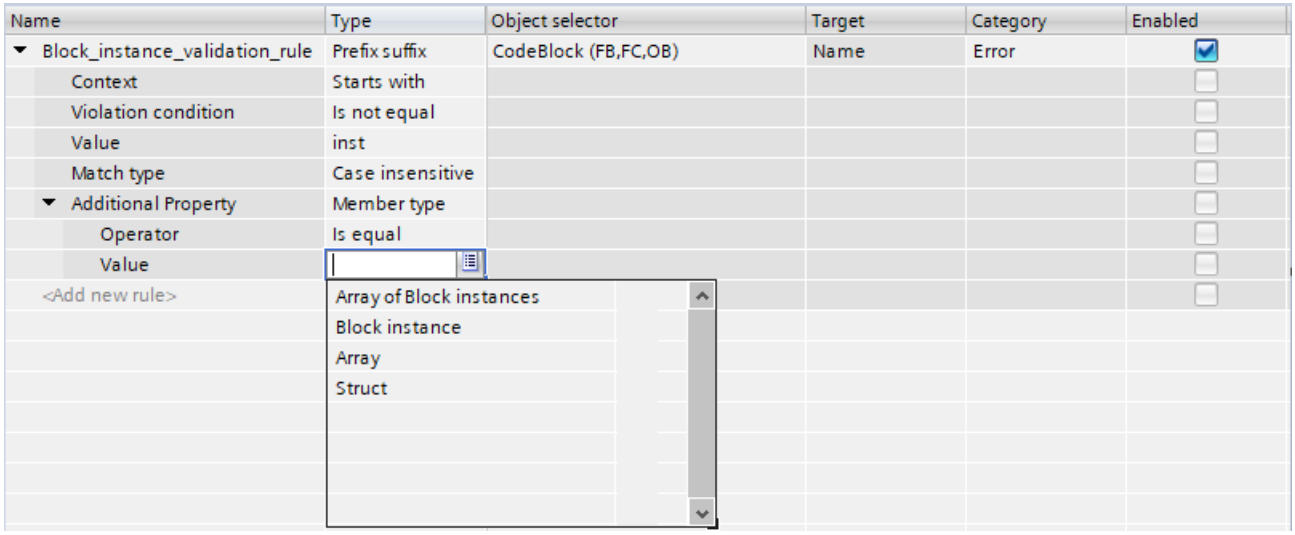
Array

For Arrays, the names such as "data" , "beltConveyors" are correct naming conventions while "datas", "beltConveyor" are incorrect naming conventions. The below image shows the additional property funtionality that can be used to identify these violations.



Block Instance

Block instance member type validates multi instance and parameter instance variables. For Block Instance , the names such as "Inst_data" , "inst_beltConveyors" are correct naming conventions while "dataInstance", "beltConveyors_instance" are incorrect naming conventions. The below image shows the additional property functionality that can be used to identify these violations.



Note

Array of Block instances validates array of multi instance.

Data type

The data type property provided to validate the rule for the below listed data types of the variables defined in the program blocks.

Binary numbers	Bool, Byte, Word, DWord, LWord
Integers	Sint, USint, Int, UInt, Dint, UDint, Lint, ULint
Floating-point numbers	Real, LReal
Timers	Time, Stime, LTime
Date and time	Date, Time_Of_Day, LTime_Of_Day, Date_And_Time, LDT, DTL
Character strings	Char, WChar, String, WString
UDT	PLC data types
Pointer	REF_TO, VARIANT, DB_ANY, POINTER, ANY
Parameter types	BLOCK_FC, BLOCK_FB, BLOCK_DB
System data types	IEC_TIMER, IEC_LTIMER, IEC_SCOUNTER, IEC_USCOUNTER, IEC_COUNTER, IEC_UCOUNTER, IEC_DCOUNTER, IEC_UDCOUNTER, IEC_LCOUNTER, IEC_ULCOUNTER

The target object selectors for which the above property will be applicable are as follows:

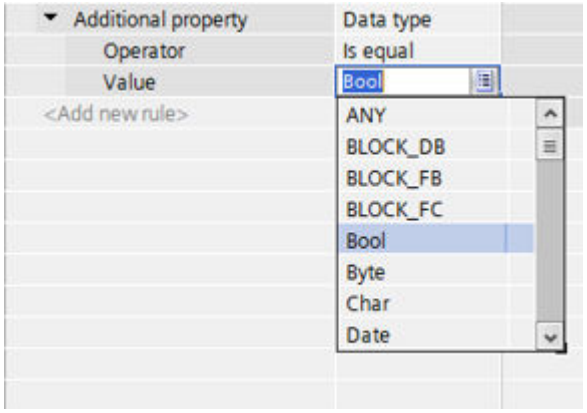
- CodeBlock.Interface (In,Out,InOut)
- CodeBlock.Interface.Input
- CodeBlock.Interface.Output
- CodeBlock.Interface.InOut
- CodeBlock.Interface.Static
- CodeBlock.Interface.Temp
- GlobalDataBlock.Static
- PLCDataType.Member

Note

- Rule execution for other object selectors will be ignored.
- This property will not be applicable to the child nodes of any Struct / Array/Multi instance / Parameter instance/UDTs.
- Multi-instance/Parameter instance is applicable for User/System FBs and PLC data types including UDTs/SDTs.

Examples

In the below image, additional property is set to Data type value can be set to any of the above mentioned data type values for identifying violations. Here we have chosen Bool type as value.



Block type

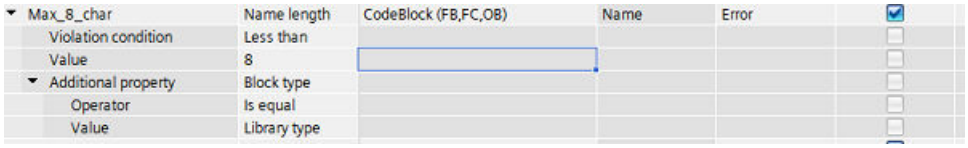
The rule definition with block type as library type additional property validates only when the block type reaches with the specified additional property. It accepts the value as library type and validates the rule only for library blocks.

Here are the possible blocks/UDTs value scenarios.

Block	Additional Property	Value
Library type FC	Block type	Library Type
Library type FB	Block type	Library Type
Library type UDT	Block type	Library Type

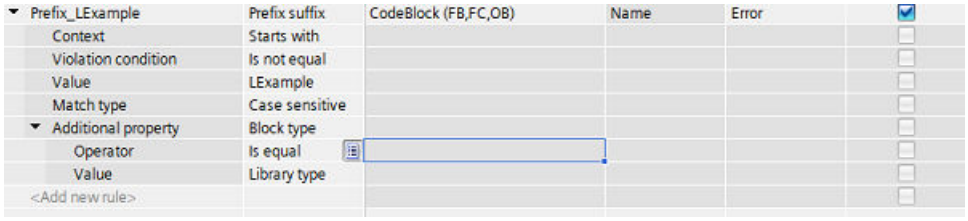
Examples

In the image, additional property is set to block type with value library type for identifying violations with name less than 8 letters.



Max_8_char	Name length	CodeBlock (FB,FC,OB)	Name	Error	☑
Violation condition	Less than				☐
Value	8				☐
Additional property	Block type				☐
Operator	Is equal				☐
Value	Library type				☐

In the image, additional property is set to block type with value library type for identifying violations with names that starts from **LExample**.



Prefix_LExample	Prefix suffix	CodeBlock (FB,FC,OB)	Name	Error	☑
Context	Starts with				☐
Violation condition	Is not equal				☐
Value	LExample				☐
Match type	Case sensitive				☐
Additional property	Block type				☐
Operator	Is equal				☐
Value	Library type				☐
<Add new rule>					☐

Application Test 2

2.1 Introduction

Introduction

Application test enables user to define test cases for the PLC program blocks of PLC 1500 with the help of SIMATIC S7-PLCSIM Advanced in “AAA” pattern. The user can perform the following:

- Arrange : Set the values of the data that is passed to the program blocks under test.
- Act : Run the user program under test with the arranged parameters with the PLC instance created locally.
- Assert : Verify that the action under test behaves as expected.

Defining a Test Case

- Test case contains a variable definition section (optional)
- Test case will have multiple test steps to perform the following actions
  - Value assignments to PLC program block variables /PLC Tags (DB members, IO-tags).
  - RUN statement: Run the PLC for n cycles or a dedicated timespan.
  - Assert statements: Compares actual values with the expected values.

Benefits

- Support of Test-Driven Development.
- Only tested code will be released before using on real machine.

2.2 Creating and Managing Test Cases

Introduction

This chapter guides you to create and manage test cases. It also assists you in performing general test case operations.

Test case Editor

The test case editor allows you to edit each test case individually. It also enables you to peform the test case operations individually for each test case.

Prerequisites

- STEP 7 Professional V19 installed.
- SIMATIC S7-PLCSIM Advanced V6.0 is installed
- Optional Package TIA Portal Test Suite Advanced V19 is installed.

Note

As an example, we will use "test case_1" throughout the Application test chapter. The user can use any name when creating a test case.

Adding a New Test case

You can define a test case which contains several rules against the PLC objects that can be validated. The test cases added are as explained below:

Procedure

1. In the Test Suite project that you have created, double click "Add new test case". A new test case "test case_1" is added to the project tree.

Note

- Test cases are sorted automatically after adding a new entry to the project tree.
- Test cases are sorted as per the precedence of unicode characters (Similar to project tree objects sorting).

Deleting a Test case

You can delete a test case from the project tree and deleting will remove the same from the project tree under the "Application test" subfolder.

Procedure

1. Under the "Project tree" area, select "Project 1 > Test Suite > Application test > test case_1".
2. Right click "test case_1 > Delete".
3. The selected test case is now deleted.

Note

"Confirm deleting" dialog box will appear after you select test case(s) to be deleted. You can choose "Yes" or "No" based on your requirements.

Renaming a Test case

You can rename a test case in the project tree. After you rename test case(s), they will be sorted in an order of character precedence similar to the program blocks folder. You can rename test case(s) using the following methods:

Procedure

1. Under the "Project tree" area, select "Project 1 > Test Suite > Application test > test case_1".
2. Right click "test case_1 > Rename".
3. Rename the test case according to your choice.

Note

- A test case name can accept a maximum of 125 unicode charaters
- When multiple test cases are selected, you can rename the last selected test case.
- All unicode characters are accepted except the following windows characters that are reserved
  - < (less than)
  - >(greater than)
  - : (colon)
  - " (double quote)
  - / (forward slash)
  - \ (backslash)
  - | (vertical bar or pipe)
  - ? (question mark)
  - * (asterisk)
  - white space

Opening a Test case Editor

You can open a test case editor and edit it separetely by the following methods:

1. Double click on the required test case.
2. Select the required test case and click "Open"
3. Select the required test case and press "enter"
4. Select the required test case, pause and use mouse click.

Note

You also have the option to open multiple test cases.

Copy/Paste a Test case from project tree

You can copy test case(s) from the project tree. This results in additon of the added test case to the respective project tree under the "Application test" subfolder with the name similar to project tree objects. It can be done in the following ways:

1. Press "Ctrl + C".
2. Click "Copy" from the context menu.
3. Click "Copy" from the main menu.
4. Tool bar icon

Note

- You can copy single/multiple test cases from the project tree.
- Test cases are sorted automatically after you perform the paste operation based on the precedence of the unicode char
- Copy/Paste operation of a test case works only witihn the same project, and not across different projects

2.3 Aliasing a Program Variable

Aliasing a PLC Variable/PLC Tags

Application test enables the user to declare an alias (reference) variable to the PLC program variables/PLC tags for the data types supported within the VAR/END_VAR section as mentioned below:

```
VAR
<alias_name> : <PLC_variable/PLC_tag> ;
END_VAR
```

It is also possible to assign a value to the alias variables in the VAR section,

```
VAR
<alias_name> : <PLC_variable/PLC_tag>  := <value> ;
END_VAR
```

Example:

```
VAR
motorStart : "SeverInterface_DB".‟InstDB.boolVar" := TRUE;
"motor Start" : "SeverInterface_DB".‟InstDB.boolVar" := TRUE;
END_VAR
```

Where, `motorStart` is the alias name for the `boolVar` which is present in `"InstDB"` of selected server interface file.

Note

Test case variables must be in double quotation marks in case the name contains special characters or space.

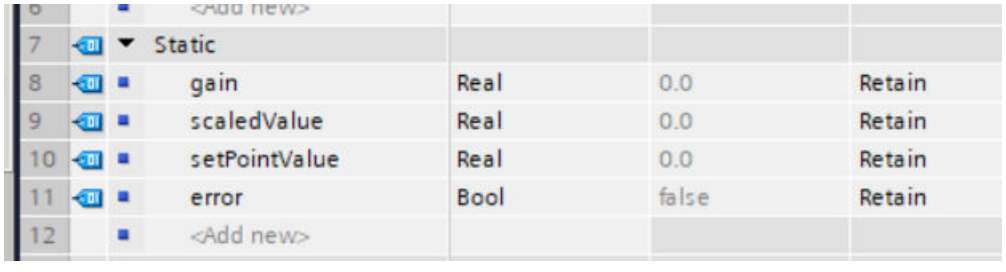
User has the access to provide alias for all the variables of the below mentioned Data blocks/PLC tags:

- Instance DB variables (Single/Multi-instance/Parameter instance)
- Global DB variables
- PLC tags

The above mentioned varieties of variables can be aliased in the test case editor with the same syntax as it is accessed in the PLC program block editor as shown in the table below:

Varities	Syntax
Simple variable	`<alias_name> : <PLC_variable/ PLC_tag> ;`
Array member	`<alias_name> : <PLC_variable[index]> ;`
Struct member	`<alias_name> : <PLC_var_struct.struct_mem> ;`
Struct Struct member	`<alias_name> : <PLC_var_struct.struct.struct_mem> ;`
Struct Array member	`<alias_name> : <PLC_var_struct.struct_mem[index]> ;`
Array of Struct member	`<alias_name> : <PLC_var_Arraystruct[index].mem> ;`
UDT member	`<alias_name> : <PLC_var_udt.udt_mem> ;`
Array of UDT member	`<alias_name> : <PLC_var_udt[index].udt_mem> ;`
UDT UDT member	`<alias_name> : <PLC_var_udt.udt_mem> ;`

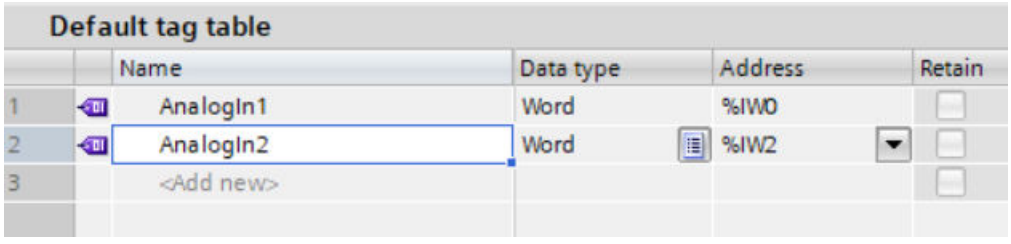
Global Sensor



Transfer Data

Static			
output	Struct		Retain
var	Array[#LOWER_BOUND..#UPPER_BOUND] of Int		Retain
var[0]	Int	0	Retain
var[1]	Int	0	Retain
var[2]	Int	0	Retain
var[3]	Int	0	Retain
var[4]	Int	0	Retain
var[5]	Int	0	Retain
var[6]	Int	0	Retain
var[7]	Int	0	Retain
var[8]	Int	0	Retain
var[9]	Int	0	Retain
var[10]	Int	0	Retain
array1	Array[0..1] of Real		Retain
array1[0]	Real	0.0	Retain
array1[1]	Real	0.0	Retain
Struct1	Struct		Retain
mem1	Int	0	Retain
mem2	Int	0	Retain
struct2	Struct		Retain
<Add new>			
Static_1	"type1"		Retain
mem1	Bool	false	Retain
<Add new>			

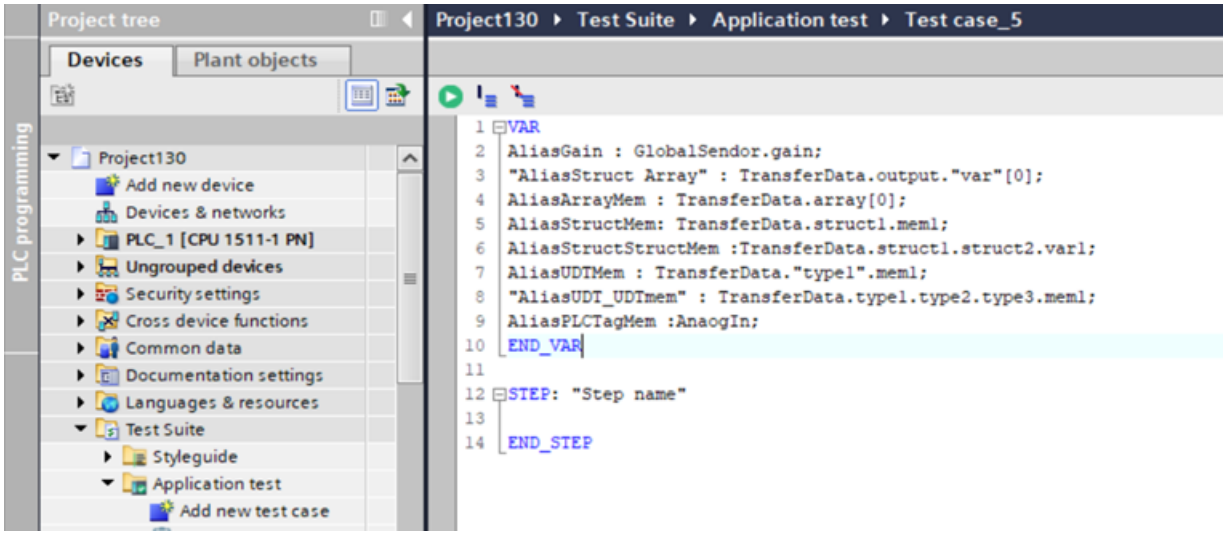
Default Tag Table



Default tag table

	Name	Data type	Address	Retain
1	AnalogIn1	Word	%IW0	
2	AnalogIn2	Word	%IW2	
3	<Add new>			

Test Case Editor Example



2.4 Basic Instructions

Basic Instructions Supported in Test Case Editor

A test case editor can have multiple test steps with the syntax as:

```
STEP : <step_name>
END_STEP
```

A step will have below mentioned statements:

Assignments

Assignment statements supported with the syntax as `"Var name"  :=  <value>;` where `"Var name"` can be a server interface variable or an alias variable.

Run

Run the PLC for a given number of cycles or a dedicated timespan

The RUN instruction will allow the user to execute the PLC program to fixed number of cycles or for a given time span dependning on the given parameter. When the user runs the test case, the PLC program is executed using PLCSIM Advanced.
Syntax: `RUN(Cycles := <value>); / RUN(Time := <value>);`

Parameters

The following table shows the parameters of the instruction:

Parameter	Data Type	Value	Description
Cycles	Unsigned Integer	Constant	Number of cycles
Time	Time	Constant	Dedicated Time Span

Note

As stated in the PLCSIM Adv manual under section API Instances > "StartProcessing()", the virtual controller will run at least the requested time.

Example: If you want to run 1ms and the first cycle is completed after 0.987ms, it means that there will be another cycle until the virtual PLC has run for at least 1ms. It cannot hit the exact point in time when 1ms is completed. There will be always a kind of jitter.

Run with condition

The `RUN` instruction can be used with an optional parameter for conditional execution of test case. Once the optional parameter reaches the user specified value or maximum number of cycles is reached, the execution is concluded.

Syntax: `RUN( Cycles := <value>, <optional_parameter> = <value>);`

Parameters

The following table shows the parameters of the instruction:

Parameter	Data Type	Value	Description
Cycles	Unsigned Integer	Constant	Number of cycles
Optional Parameter	Integer, Boolean, Byte, Word	Constant	Value of the Optional Parameter

Examples

Conveyor block: A conveyor length and its maximum speed can vary and the time taken by a product from A to B can change significantly. Therefore, the limited variations of the Run statements (number of CPU cycles / Time) is not sufficient for such applications. Currently the user would have to create different test cases for different conveyors or always specify the worst case scenario in terms of CPU cycles and time only.

Multifunctional blocks: The execution of several function blocks depends on the other in cases when output1 on block A comes TRUE, then block B can be executed, and so on. Therefore the optional parameter can represent the status of block and used to specify the run condition instead of cycles or time and react to specific values.

Assert Statements

The ASSERT statements, compare the actual value of the server interface variable with the expected value specified by the user.
Syntax: `Assert.<_Condition_>(<actual_value>,<expected value>);`

Parameters

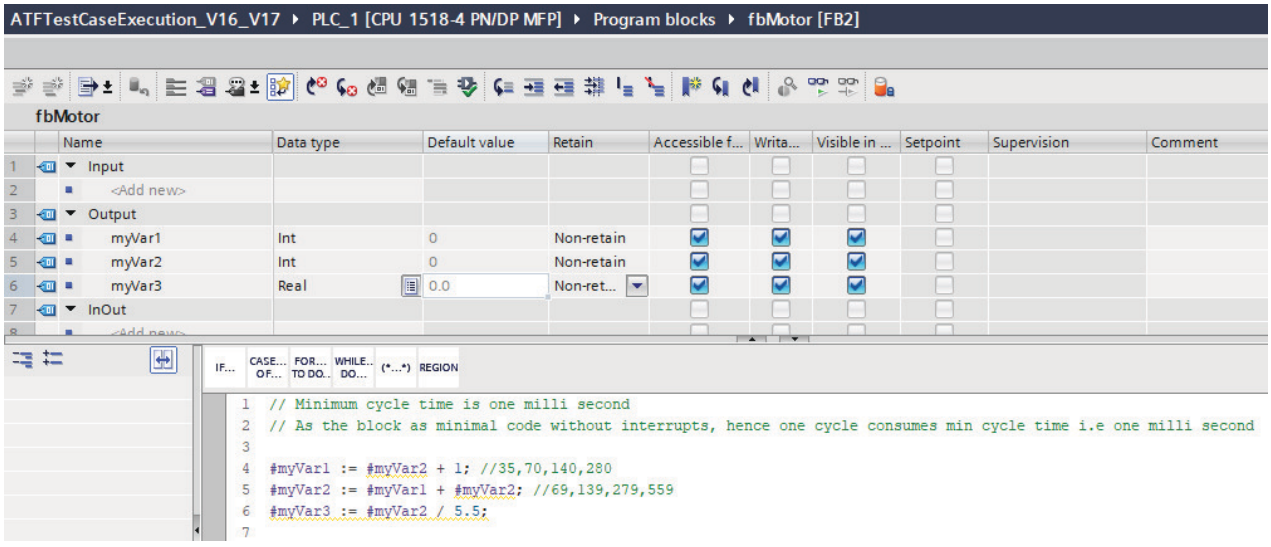
The following table shows the parameters of the instruction:

Instruction	Parameters	Description	Data type
Assert.Equal()	actual_value, expected_value	Assert.Equal validates whether the actual value is equal to expected value	Integers /Binary numbers / Floating point numbers/ Char/Time
Assert.NotEqual()	actual_value, expected_value	Assert.NotEqual validates whether the first parameter is not equal to expected value	Integers /Binary numbers / Floating point numbers/ Char/Time
Assert.GreaterThan()	actual_value, expected_value	Assert.GreaterThan validates whether the first parameter is greater than the expected value	Integers /Binary numbers / Floating point numbers/ Char/Time except BOOL datatype
Assert.GreaterThanOrEqual()	actual_value, expected_value	Assert.GreaterThanOrEqual validates whether the first parameter is greater than or equal to expected value	Integers /Binary numbers / Floating point numbers/ Char/Time except BOOL datatype
Assert.LessThan()	actual_value, expected_value	Assert.LessThan validates whether the first parameter is less than the expected value	Integers /Binary numbers / Floating point numbers/ Char/Time except BOOL datatype
Assert.LessThanOrEqual()	actual_value, expected_value	Assert.LessThanOrEqual validates whether the first parameter is less than or equal to expected value	Integers /Binary numbers / Floating point numbers/ Char/Time except BOOL datatype
Assert.InRange()	actual_value, expected_min_value, expected_max_value	Assert.InRange helps to compare the expected value with actual value using minimum and maximum values defined	Integers /Floating point numbers

Note

For validating floating point numbers using Assert.InRange() is recommended.

Example:



Test case for the above PLC program data:

```
VAR
tcVar1 : fbMotor_DB.myVar1;
tcVar2 : fbMotor_DB.myVar2;
tcVar3 : fbMotor_DB.myVar3;
END_VAR
```

```
STEP : "Test step to check the status after given cycles"
tcVar2 := 34;
RUN(Cycles := 4);
Assert.Equal(tcVar1,280);
Assert.Lessthan(tcVar2,560);
Assert.Greaterthan(tcVar2,550);
END_STEP
```

```
STEP : "Test step to check the status after given time"
tcVar2 := 34;
RUN(Time := T#4ms);
Assert.Equal(tcVar2,280);
Assert.Lessthan(tcVar2,560);
Assert.Greaterthan(tcVar2,550);
END_STEP
```

```
STEP : "Test step to check the status after given cycles or when optional parameter reaches
the specified value"
tcVar1 := 100;
RUN(Cycles := 10, tcVar1 = 1000);
Assert.Equal(tcVar1,300);
Assert.Lessthan(tcVar1,500);
Assert.Greaterthan(tcVar1,200);
END_STEP
```

```
STEP : "Test step to check the status after given time"
RUN(Time := T#4ms);
//The below Assert statement will PASS, if tcVar3 is less than 1.009 and greater than 1.005
Assert.InRange(tcVar3,1.005,1.009);
END_STEP
```

Assert with custom description

You can use the optional parameter to add any information about the jobs verified using Assert statements. This option is available for all Assert types.

Syntax:

```
Assert.<_Condition_>(<actual_value>,<expected value>,<Comments>);
```

```
Assert.<_Condition_>(<actual_value>,<expected
value>,<Comments>+&<variable_name>+<Comments>);
```

Note

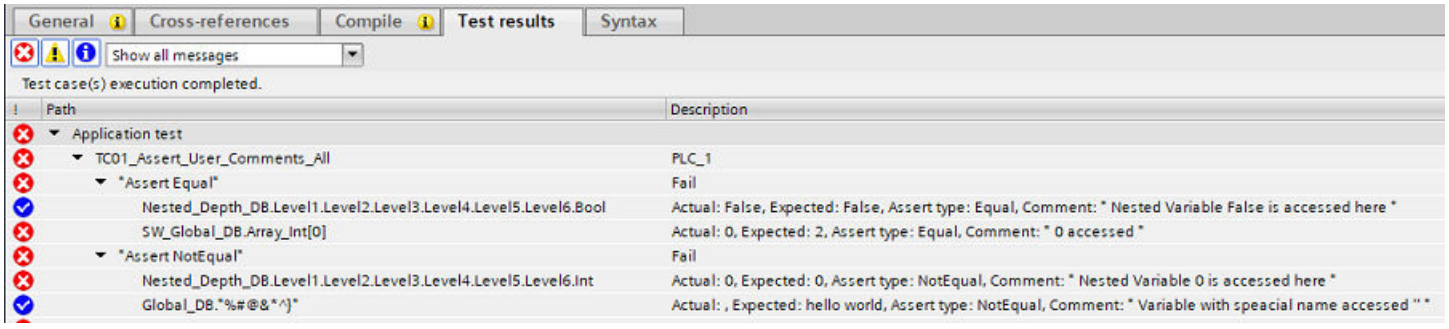
- You can query state of the any other signals using "&" as a notation in custom description of the Assert statements.
- You will be able to query the signals of all the supported data types. [See Supported Data Types (Page 49)]

Example:

```
VAR
alias_Nested_Int:"Nested_Depth_DB".Level1.Level2.Level3.Level4.Level5.Level6.Int;
END_VAR

STEP: "Assert Equal"
RUN(CYCLES:=2);
ASSERT.Equal("Nested_Depth_DB".Level1.Level2.Level3.Level4.Level5.Level6.Bool,0,"Nested
Variable"+&"Nested_Depth_DB".Level1.Level2.Level3.Level4.Level5.Level6.Bool + "is accessed
here");
ASSERT.Equal("SW_Global_DB".Array_Int[0],16#2,&"SW_Global_DB".Array_Int[0] + "accessed");
END_STEP

STEP: "Assert NotEqual"
RUN(CYCLES:=2);
ASSERT.NotEqual(alias_Nested_Int,0,"Nested Variable"+&alias_Nested_Int + "is accessed
here");
ASSERT.NotEqual("Global_DB"."%#@&*^}",'hello world',"Variable with speacial name
accessed"+&"Global_DB"."%#@&*^}");
END_STEP
```



Copying Code Examples Into the Program

To copy a code example:

1. Open the help topic with the sample program.
2. Click "Copy" in the header of the program example.
3. The program code is copied to the clipboard.
4. Open the programming edtior and insert the copied code. Select "Paste" in the shortcut menu. The copied program code is inserted from the clipboard into the program code edited.

2.5 Supported Data Types

2.5.1 Binary Numbers

Binary Numbers

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Bool	Boolean	FALSE or TRUE	FALSE or TRUE	FALSE or TRUE
	Unsigned integers (decimal system)	0 or 1	0 or 1	
	Binary numbers	2#0 or 2#1	2#0 or 2#1	
	Hexadecimal numbers	16#0 or 16#1	16#0 or 16#1	
Byte	Integers (decimal system)	Signed integers: -128 to +127 Unsigned integers: 0 to 255	15	16#0 to 16#FF
	Binary numbers	2#0 to 2#1111_1111	2#00001111	
	Hexadecimal numbers	16#0 to 16#FF	16#0F	
Word	Integers (decimal system)	Signed integers: -32_768 to +32_767 Unsigned integers: 0 to 65_535	61680	16#0 to 16#FFFF
	Binary numbers	2#0 to 2#1111_1111_1111_1111	2#1111000011110000	
	Hexadecimal numbers	16#0 to 16#FFFF	16#F0F0	
Dword	Signed integers (decimal system) Unsigned integers (decimal system)	-2_147_483_647 to +2_147_483_647. 0 to 4_294_967_295	0	16#0000_0000 to 16#FFFF_FFFF
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111	2#00000000111100001111111100001111	
	Hexadecimal numbers	16#0000_0000 to 16#FFFF_FFFF	16#00F0FF0F	

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Lword	Integers (decimal system)	Signed integers: -9_223_372_036_854_775_808 to +9_223_372_036_854_775_807 Unsigned integers: 0 to 18_446_744_073_709_551_615	+26_123_590_360_715	16#0000_0000 to 16#FFFF_FFFF_FFFF_FFFF
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111	2#0000_0000_0000_0000_0000_1011_1110_0001_0010_1111_0101_0010_1101_1110_1000_1011	
	Hexadecimal numbers	16#0000_0000 to 16#FFFF_FFFF_FFFF_FFFF	16#0000_0000_5F52_DE8B	

2.5.2 Integers

Integers

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Sint	Signed integers (decimal system)	-128 to +127	44	-128 to +127
	Binary numbers (only positive)	2#0 to 2#0111_1111	2#0010_1100	
	Hexadecimal numbers (only positive)	16#0 to 16#7F	16#2C	
Int	Signed integers (decimal system)	-32_768 to +32_767	3_785	-32_768 to +32_767
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111	2#0000_1110_1100_1001	
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF	16#0EC9	

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Dint	Signed integers (decimal system)	-2_147_483_648 to +2_147_483_647	+125_790	-2_147_483_648 to +2_147_483_647
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111_1111_1111_1111_1111	2#0000_0000_0000_0001_1110_1011_0101_1110	
	Hexadecimal numbers	16#0 to 16#7FFF_FFFF	16#0001_EB5E	
Usint	Unsigned integers (decimal system)	0 to 255	78	0 to 255
	Binary numbers	2#0 to 2#1111_1111	2#0100_1110	
	Hexadecimal numbers	16#0 to 16#FF	16#4E	
Uint	Unsigned integers (decimal system)	0 to 65_535	65_295	0 to 65_535
	Binary numbers	2#0 to 2#1111_1111_1111_1111	2#1111_1111_0000_1111	
	Hexadecimal numbers	16#0 to 16#FFFF	16#FF0F	
Udint	Unsigned integers (decimal system)	0 to 4_294_967_295	4_042_322_160	0 to 4_294_967_295
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111	2#1111_0000_1111_0000_1111_0000_1111_0000	
	Hexadecimal numbers	16#0 to 16#FFFF_FFFF	16#F0F0_F0F0	
Lint	Signed integers (decimal system)	-9_223_372_036_854_775_808 to +9_223_372_036_854_775_807	+154_325_790_816_159	-9_223_372_036_854_775_808 to +9_223_372_036_854_775_807
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111	2#0000_0000_0000_0000_1000_1100_0101_1011_1100_0101_1111_0000_1111_0111_1001_1111	
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF_FFFF_FFFF_FFFF	16#0000_8C5B_C5F0_F79F	

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Ulint	Unsigned integers (decimal system)	0 to 18_446_744_073_709_551_615	154_325_790_816_159	0 to 18_446_744_073_709_551_615
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111	2#0000_0000_0000_0000_1000_1100_0101_1011_1100_0101_1111_0000_1111_0111_1001_1111	
	Hexadecimal numbers	16#0 to 16#FFFF_FFFF_FFFF_FFFF	16#0000_8C5B_C5F0_F79F	

2.5.3 Floating Point Numbers

Floating Point Numbers

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Real	Floating-point numbers according to IEEE754	-3.402823e+38 to -1.175495e-38	1.0e-5;	Only one type format supported as mentioned in value range
		±0.0		
	Floating-point numbers	+1.175495e-38 to +3.402823e+38	1.0;	
Lreal	Floating-point numbers according to IEEE754	-1.7976931348623157e+308 to -2.2250738585072014e-308	1.0e-5;	Only one type format supported as mentioned in value range
		±0.0		
	Floating-point numbers	+2.2250738585072014e-308 to +1.7976931348623157e+308	1.0;	

2.5.4 Timers

Timers

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Time	Signed duration	T#-24d_20h_31m_23s_648ms to T#+24d_20h_31m_23s_647ms	T#10d_20h_30m_20s_630ms, TIME#10d_20h_30m_20s_630ms	T#-24d_20h_31m_23s_648ms to T#+24d_20h_31m_23s_647ms

2.5.5 Character Strings

Strings

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Char	CHAR	ASCII character set ASCII Values (32 - 127)	'A'(Characters which represents the ASCII value 32-127)	'Character e.g. 'A'
WChar	Unicode,ASCII Values	$0000 - $D7FF	'WCHAR#'a'	
String	ASCII character string incl. special characters	0 to 254 characters	'Name' STRING#'NAME' STRING#'Na... (The actual length of the string is longer than the space available on the screen.) STRING#'' (The string is empty.)	STRING#'NAME'
WString	Unicode character string; n specifies the length of the character string.	Preset value: 0 to 254 characters Max. possible value: 0 to 16382	'WSTRING#'Hello World' WSTRING#'Hello Wo... (The actual length of the string is longer than the space available on the screen.) WSTRING#'' (The string is empty.)	WSTRING#'Hello World'

2.6 Test Case Execution

2.6.1 Introduction

Introduction

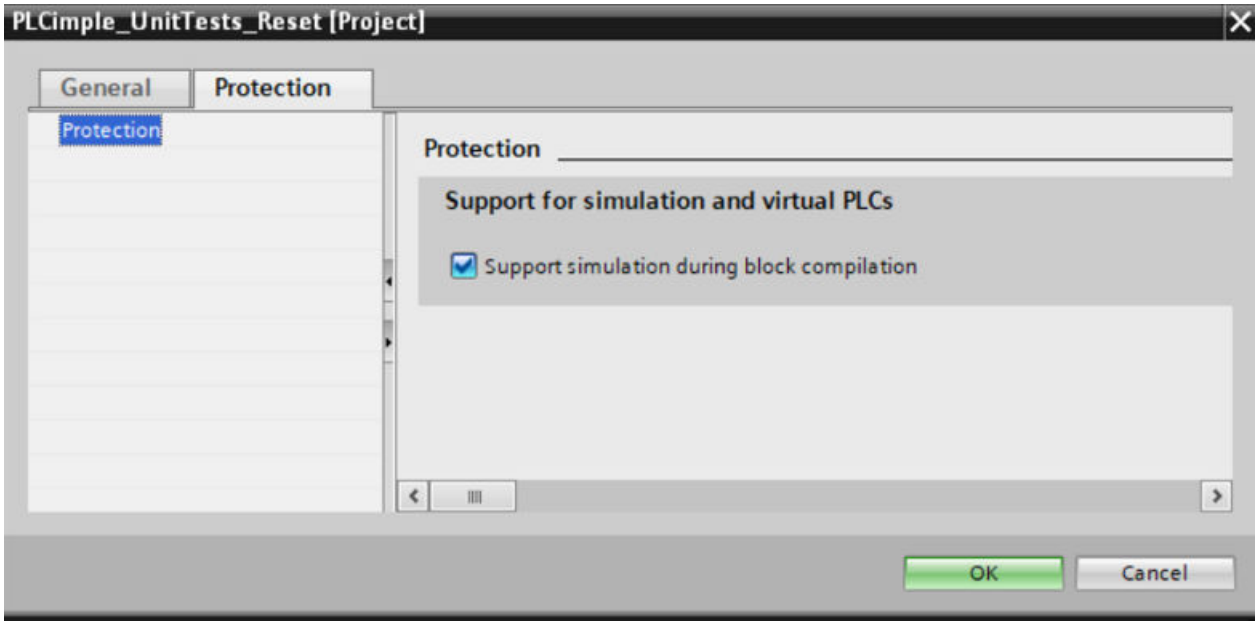
You can run the tests defined for a given S7-application using the below modes

- SystemManagedPLCSIM Instance
- ExternallyManagedPLCSIM Instance

To run the test cases, you must install TIA Portal Test Suite together with SIMATIC S7-PLCSIM Advanced. Hence, the following restrictions of PLCSIM Advanced must be considered:

- Simulation support must be enabled in the project properties.
- It is recommended that the PLCSIM Advanced requirements are met before using Application test as the results may vary significantly.

For more information, refer to PLCSIM Advanced manual.



Supported CPUs

Test case execution can be done on the selected PLC in the scope. The scope contains only PLC 1500 which are compatible with PLCSIM Advanced as mentioned below:

Type	Firmware version V1.8 to V3.0	
	Standard	**Fail-safe**
Standard CPUs	CPU 1511-1 PN CPU 1513-1 PN CPU 1515-2 PN CPU 1516-3 PN/DP CPU 1517-3 PN/DP CPU 1518-4 PN/DP CPU 1518-4 PN/DP ODK CPU 1518-4 PN/DP MFP	CPU 1511F-1 PN CPU 1513F-1 PN CPU 1515F-2 PN CPU 1516F-3 PN/DP CPU 1517F-3 PN/DP CPU 1518F-4 PN/DP CPU 1518F-4 PN/DP ODK CPU 1518F-4 PN/DP MFP
Compact CPUs	CPU 1511C-1 PN CPU 1512C-1 PN	-
ET 200SP CPUs	CPU 1510SP-1 PN CPU 1512SP-1 PN CPU 1514SP-2 PN* CPU 1514SP T-2 PN*	CPU 1510SP F-1 PN CPU 1512SP F-1 PN CPU 1514SP F-2 PN* CPU 1514SP TF-2 PN*
Technology CPUs	CPU 1511T-1 PN CPU 1515T-2 PN CPU 1516T-3 PN/DP CPU 1517T-3 PN/DP CPU 1518T-4 PN/DP*	CPU 1511TF-1 PN CPU 1515TF-2 PN CPU 1516TF-3 PN/DP CPU 1517TF-3 PN/DP CPU 1518TF-4 PN/DP*
R/H-CPUs	CPU 1513R-1 PN CPU 1515R-2 PN CPU 1517H-3 PN	CPU 1518HF-4 PN
ET 200pro CPUs	CPU 1513pro-2 PN CPU 1516pro-2 PN	CPU 1513pro F-2 PN CPU 1516pro F-2 PN

Type	Firmware version V1.8 to V3.0	
	Standard	**Fail-safe**
SIMATIC Drive Controller	-	CPU 1504D TF CPU 1507D TF
*S7-1500 Software Controller	CPU 1505SP CPU 1505SP T CPU 1507S CPU 1508S	CPU 1505SP F CPU 1505SP TF CPU 1507SP F CPU 1508S F

*These CPUs are not supported for SystemManagedPLCSIMInstance mode of Test case execution

2.6.2 SystemManagedPLCSIM Instance

SystemManagedPLCSIM Instance

To run a test case in this mode, you must install TIA Portal Test Suite together with SIMATIC S7-PLCSIM Advanced V6 on the same PC

Execution of a test case will automatically perform the below mentioned steps without user interaction:

- Creates PLCSIM Advanced Instance.
- Compiles and download the specific PLC selected in the test case scope from the project to PLCSIM Advanced instance
- Executes a test case
- Deletes PLCSIM Advanced Instance which was created automatically.
- Displays test results in the inspector window
- Saves test results under Common data\Logs

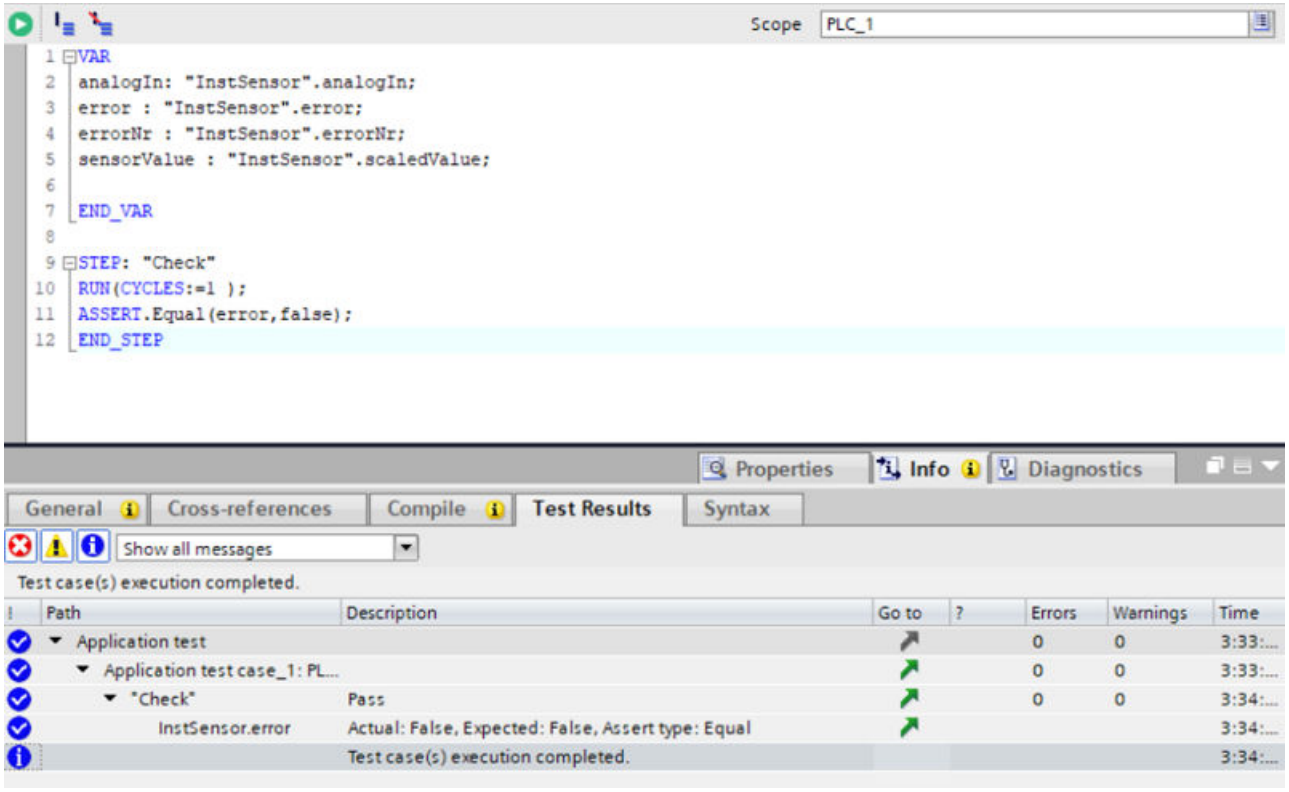
PLCSIM Advanced license is not required to run the tests in this mode of execution.

The test execution can be started in two different ways, either from the current opened test case editor or with a context menu entry on the test case in the project tree.

Execution of a test case(s) will include the following steps:

- The PLC program will be compiled and checked for consistency.
- If the compilation fails, then the execution is stopped.
- If the compilation is successful, then the compiled PLC program will be downloaded to the simulated PLC device by creating an instance in PLCSIM Advanced.
- After the successful test case execution, the instance created will be deleted from PLCSIM Advanced.

- Results will be displayed in the "Test Results" tab in the inspector window with 'Pass'/'Fail' feedback for each test step and the 'Go to" option will be provided to navigate to the specific test step.
- Log file will be created for each test case execution under common data\Logs.



Note

- During test case execution, the user will not be able to use TIA Portal.
- You can cancel the test case execution at any point of time. But, the cancellation status is displayed only after the current test case step execution is complete.
- Working with Safety blocks always requires to introduce a test step with at least RUN() statement, before working with Fail-safe IOs in the next test step. This is a system behavior for Fail-safe IOs. It is expected that the PLC should be in RUN mode for at least one safety cycle before writing values to F-Inputs.

How to work with PLCSIM Advanced V6.0?

Possible causes of error:

It is not possible to run the test(s) in secure mode using PLCSIM Advanced V6.0.

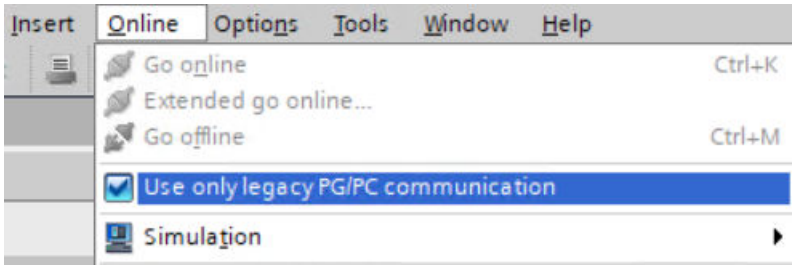
Solution:

Enable the legacy mode before starting the test case execution.

Enabling legacy mode:

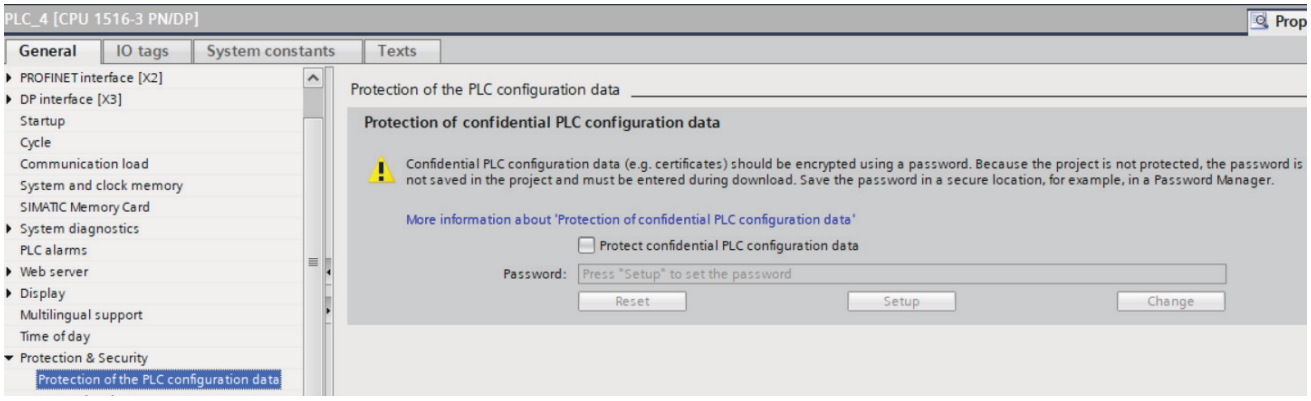
To enable the legacy mode in the project, proceed as follows:

1. In the "Online" menu, select the "Use only legacy PG/PC communication" check box.



For more information, refer "Additional settings for the secure PG/HMI communication".

In addition to test with PLC FW V2.9 and higher, the option "Protect confidential PLC configuration data" needs to be disabled.



2.6.3 ExternallyManagedPLCSIM Instance

ExternallyManagedPLCSIM Instance

With Application test, you can also validate the tests written for S7-application using pre-configured S7-PLCSIM Advanced instance.

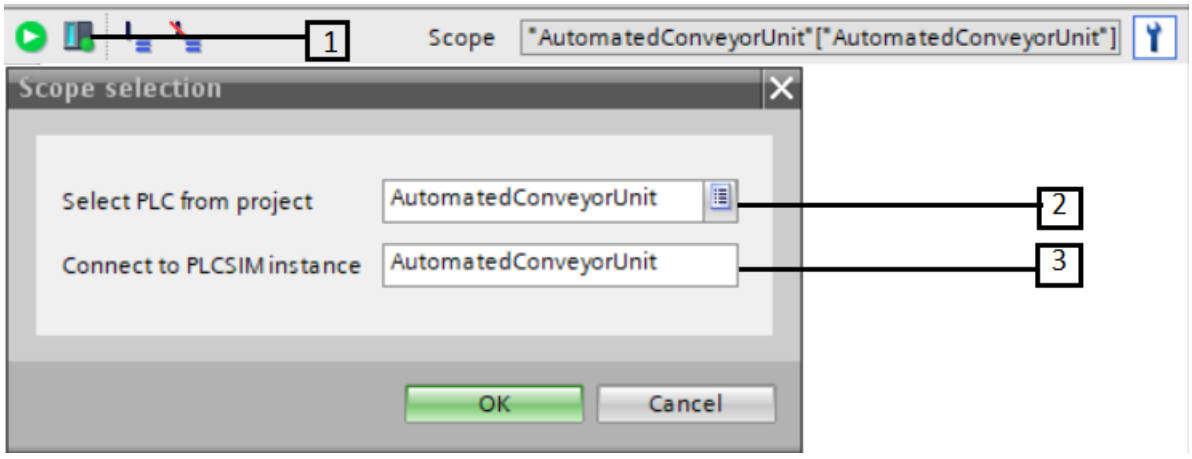
Benefits

- PLCSIM Advanced instance only need to be prepared once as it is not required to compile and download the selected S7-program for every test case execution. Therefore, a lot of time can be saved during creation and testing of new test cases and executing CI pipelines with multiple tests.
- More flexibility to create dependent and independent test cases by using pre-configured PLCSIM Advanced instance for all tests. With the new INIT/END_INIT section user can define, if a memory reset needs to be performed before starting a new independent test case. Please see the "ExternallyManagedPLCSIM Instance mode for independent/dependent tests" section for more details.

Prerequisites to run the test cases in this mode

- A valid TIA Portal Test Suite license must be available.
- S7-PLCSIM Advanced (V4.0 and higher) should be installed in the local machine.
- Create a S7-PLCSIM Advanced instance and download the up-to-date target PLC program you want to test.
- S7-PLCSIM Advanced instance details should be available in test case scope to run the test case successfully.

You can provide these details using the test case Scope selection as shown below.



1. **Externally managed PLCSIM Instance mode** : Select the mode as "ExternallyManagedPLCSIMInstance" mode to run the test case with the pre-configured S7-PLCSIM Advanced instance.
2. **Select PLC from project**: This option helps you to select the target project PLC configured in the TIA Portal. By selecting this option, you will get an intellisense support which list the PLC variables available to write the tests.
3. **Connect to PLCSIM instance**: The name of the user pre-configured S7-PLCSIM Advanced instance, where you want to run the test cases.

How it works?

The following steps will be performed when you run the test cases with the existing S7-PLCSIM Advanced instance.

- Connects to the user pre-configured S7-PLCSIM Advanced instance as mentioned in the test case scope.
- Runs the test cases on this instance.
- Results will be displayed in the "Test Results" tab in the inspector window with 'Pass'/'Fail' feedback for each test step and the 'Go to" option will be provided to navigate to the specific test step.
- Log file will be created for each test case execution under Common data\Logs.
- Above steps will be repeated for all the steps in the sequence.

Points to note

- It is possible to run the test cases only when the S7-PLCSIM Advanced instance is in "Run" mode.
- PLCSIM Advanced instance will be in "Freeze" state while writing(Assignments) and reading(Asserting) the variable values.
- At the end of every test case execution, instance will be switched back to "Run" mode.
- Test results with "ExternallyManagedPLCSIMInstance" and "SystemManagedPLCSIMInstance" will not be exactly same due to S7-PLCSIM Advanced timespan mode behavior but this is not the case with Run in Cycle mode.
- It is not possible to run the test case in case the instance is marked as "Error" due to runtime error after downloading the program.
- It is not possible to run the test case when the instance is in "HOLD" state.
- Always make sure to configure right "Project PLC" (Used to write the test case) and "PLCSIM instance" (Used to run the test case) to avoid unnecessary errors during test case execution.
- It is possible to terminate the test case execution only after performing the current action which is in progress (Assignment/Assert/Run).
- PLC variable of data type 'String' cannot be read/written with a single char value such as 'a','S','1' etc.

ExternallyManagedPLCSIM Instance mode for independent/dependent tests

Application test support a new section "`INIT/END_INIT`" , this is an optional section and can be used before `VAR/END_VAR` section in order to perform a memory reset before starting execution of a test case as shown.

```
INIT
PLC_MEMORY_RESET := TRUE; // Perform Memory reset
END_INIT
```

```
VAR
  fInput1 : "UnitTestDb".sMuxReal.fInput1;
  fInput2 : "UnitTestDb".sMuxReal.fInput2;
  fInput3 : "UnitTestDb".sMuxReal.fInput3;
  fInput4 : "UnitTestDb".sMuxReal.fInput4;
  fInput5 : "UnitTestDb".sMuxReal.fInput5;
  fInput6 : "UnitTestDb".sMuxReal.fInput6;
  fInput7 : "UnitTestDb".sMuxReal.fInput7;
  fInput8 : "UnitTestDb".sMuxReal.fInput8;
  bSelect1 : "UnitTestDb".sMuxReal.bSelect1;
  bSelect2 : "UnitTestDb".sMuxReal.bSelect2;
END_VAR
STEP: "Step name"  //Reinitialize the multiplexer inputs with new values
    fInput1 := 1100.0;
    fInput2 := 1200.0;
    fInput3 := 1300.0;
    fInput4 := 1400.0;
    fInput5 := 1500.0;
    fInput6 := 1600.0;
    fInput7 := 1700.0;
    fInput8 := 1800.0;
    bSelect1 := 0;
    bSelect2 := 0;
RUN(CYCLES:= 1);
END_STEP
```

About PLC_MEMORY_RESET

When you set `PLC_MEMORY_RESET` as TRUE then the test case execution will be started after performing memory reset on the selected PLCSIM Advanced instance else the test case execution will started with the previous values available in the virtual controller.

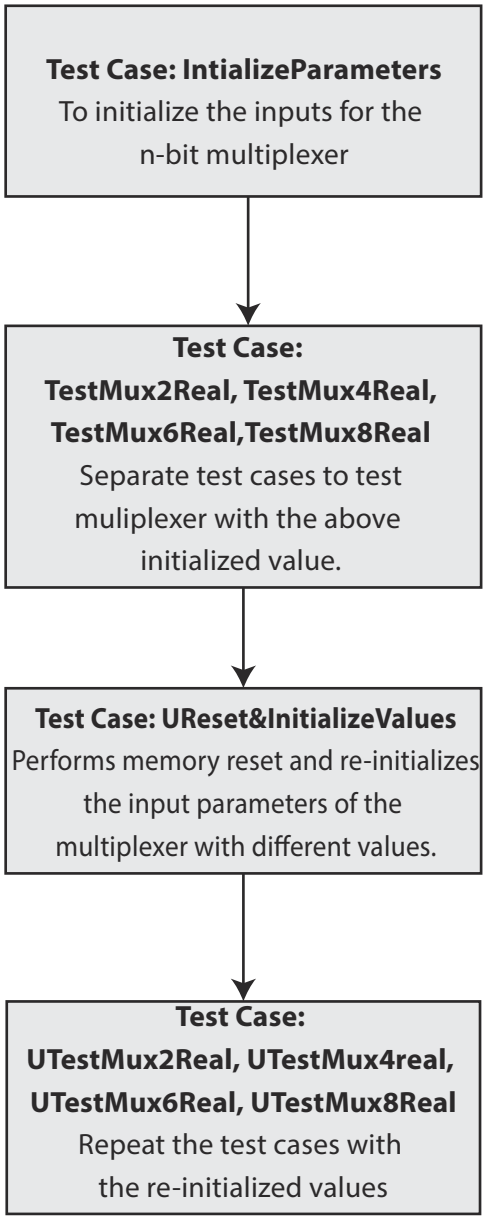
When you use INIT section with "`PLC_MEMORY_RESET` to true " then before executing the test case memory reset will be performed on the selected PLCSIM Advance instance.
The below actions are performed for "Memory reset"

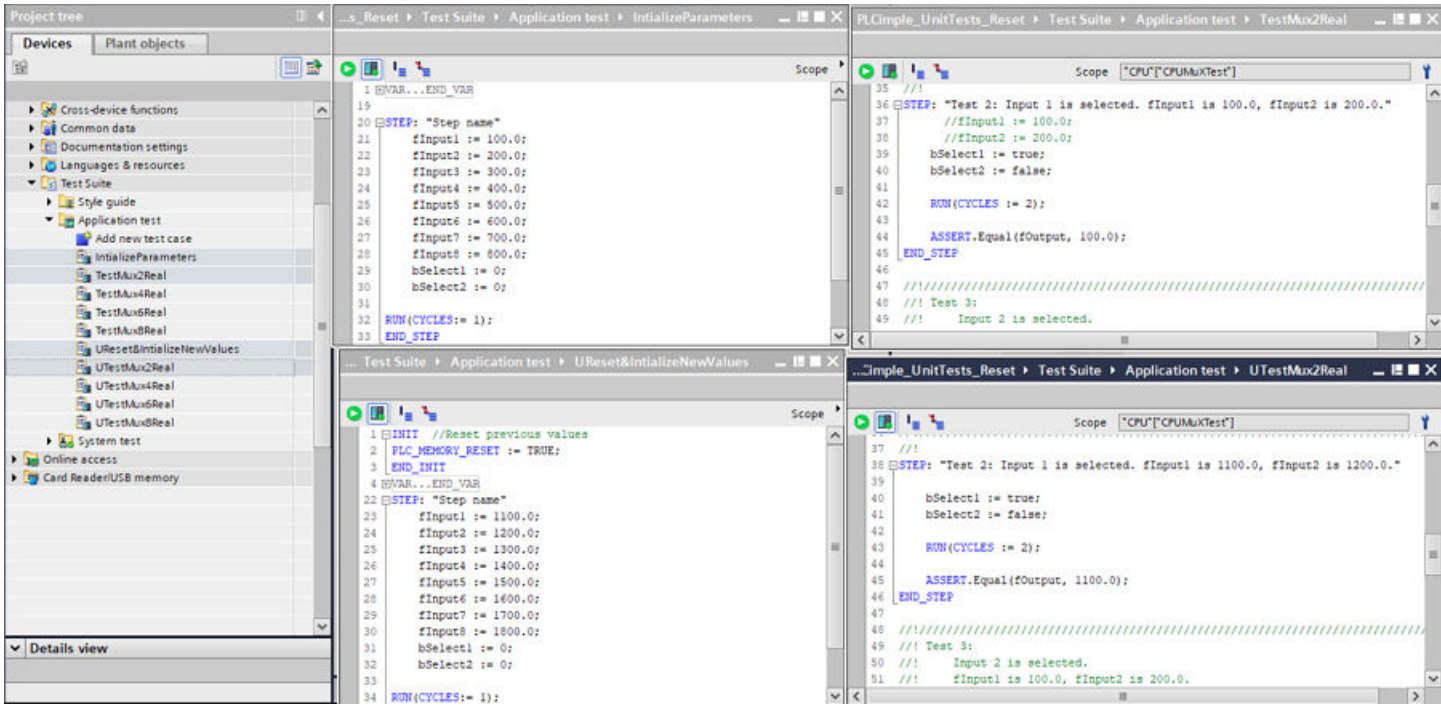
- Shuts down the virtual controller.
- Closes its processes.
- Performs a restart.
- Test case execution will be performed on the instance restarted.

Example

To test an s7-application for n-bit multiplexer the below test cases are created to test n-bit multiplexer. Each block in the diagram is represented by its corresponding test case:

- **Test case 1** : To initialize the inputs for the n-bit multiplexer
- **Test case 2 to Test case 5** : Separate test case to test multiplexer with the above initialized values.
- **Test case 6** : Performs Memory reset and reinitializes the input parameters of multiplexer with different values.
- **Test case 7 to Test case 10** : Repeat the test cases with reinitialized values in Test case 6.


Test Case: IntializeParameters
To initialize the inputs for the n-bit multiplexer
Test Case: TestMux2Real, TestMux4Real, TestMux6Real,TestMux8Real
Separate test cases to test muliplexer with the above initialized value.
Test Case: UReset&InitializeValues
Performs memory reset and re-initializes the input parameters of the multiplexer with different values.
Test Case: UTestMux2Real, UTestMux4real, UTestMux6Real, UTestMux8Real
Repeat the test cases with the re-initialized values



2.7 Test Case Execution Failure

This section helps you in evaluating reasons for test case execution failure..

If the following preconditions are not met, then you will not be able to run the test cases.

- PLCSIM Advanced V6.0(SystemManagedPLCSIM Instance mode) should be installed in the target.
- User should select the valid Project PLC as a scope.
- Test case editor should be error free.
- Project property “Support simulation during block compilation” should be selected.
- Program blocks from the target Project PLC should be error free.

Exceptions

The below error codes are provided in ExternallyManagedPLCSIMInstance mode.

Scenario	Exception code with error message
Unable to connect as mentioned instance is not available	Failed to establish connection. Error code : TestSuite_PLCSIMInstanceNotAvaliable
Conflict while connecting the server	Failed to establish connection. Error code : TestSuite_PLCSIMInstanceConnectConflict

Scenario	Exception code with error message
Instance shutdown	Failed to establish connection. Error code : TestSuite_PLCSIMInstanceShutDown
Unable to connect as mentioned instance in "Error"/"STOP"state	Failed to establish connection. Error code : TestSuite_PLCSIMInstanceErrorState Error code : TestSuite_PLCSIMInstanceStopState
Unable to read the value.	Possible cause : Accessed data point is not downloaded to the target PLCSIM Advanced Instance Error code : TestSuite_FailedToRead

In addition, error codes from PLCSIM Advanced can bee seen when the reading/writing the value to the PLC variable is failed and the instance is not in “Run” to start the test case execution.

Please refer to PLCSIM Advanced manual for the error codes from PLCSIM Advanced.

Application test cannot establish the communication with PLCSIM Advance in the following conditions:

- Creation of instance on PLCSIM Advance fails in the following cases:
  - When you use unsupported characters for instance names (Example: <, >, :, /, \., |, ?, *, =, ., space)
  - An instance already exists on the PLCSIM Advance with the same name/IP address
- If the instance is deleted in PLCSIM Advance.
- If the PLC is in ERROR state.
- Unable to initialize the instance.
- Unable to power on the PLC.
- If the program data has huge number of DBs (Approximately 500000 entries (not PLC tags)).
- When you execute a test case with Motion control technology objects such as To_SpeedAxis,To_PositioingAxis. et

2.8 Additional Features

Introduction

Application supports the following features:

1. Master copy support
2. Import/Export of test cases

Master Copy Support

Master copy support is used to transmit application tests to other projects. It is possible to create a master copy out of an application set. Master copies are also used to create standardized copies of frequently used application sets.You can create as many items as needed and then insert them into the project.

You can store master copies either in the project library or in a global library. Master copies in the project library can only be used within the project. When you create a master copy in a global library, it can be used in different projects.

Limitations of Master Copy Support

- Drag and drop of Test Suite library objects into the project tree is not supported. Therefore, use standard copy and paste instead.
- Copy and paste of user folders from library to Test Suite project tree below is not supported.

Import/Export of test cases

You can import and export the test case(s) added in the editor to an external textual file with the following options:

- **Export**: Context menu entries "Export test case(s)" in project tree when you select Application test folder/test case(s).
- **Import**: Context menu entries "Import test case(s)" in project tree when you select Application test folder.

Test case(s) can be imported/exported to an external source file with .TAT extension.

Sample of .tat textual file:

```
TEST_CASE "SingleStep"


PROPERTY
AUTHOR : "Harish"
VERSION : "0.1"
COMMENT : "This testcase contains single step"
SCOPE : "PLC1"
END_PROPERTY


VAR
"a" : "Data_block_1".varBool:=1;
"b" : "Data_block_1".varInt:=1;
END_VAR
```

```
STEP: "Step name"
RUN(Cycles:=1);
ASSERT.Equal("a",1);
ASSERT.Equal("Data_block_1".varInt,1);
END_STEP

END_TEST_CASE
```

Errors and Warnings messages for import operation:

You will also be shown status information, warnings and error messages in the Inspector window under the "Info > General" tab.

Below are the different feedback messages provided for import operation:

- Import of test case(s) completed with warnings, when:
  - The test case property 'Author' imported with the value which is not valid (Example: more than 125 characters/only space allowed).
  - The test case property 'Version' imported with the value which is not valid (Example: other than range 0.0 to 99.99).
  - The test case SCOPE imported as NULL when the scope in the file is not available in the project.
- Import of test case(s) completed with errors, when:
  - TEST_CASE keyword is missing in the textual file.
  - Test case name is missing in the textual file.
  - The test case name not enclosed in double quotes in textual file.
  - The test case name contains the characters which are not allowed (Example: <, >, :, ", /, \, |, ?, *, space) in textual file.
  - The test case name contains only NULL/Empty string in textual file.
  - The END_TEST_CASE keyword missing in textual file.

When you abort the import operation then the test case which is already in progress to import will be terminated immediately.

Multiuser engineering

Application test supports modification and execution of rule sets in Multiuser engineering environment.

Note

- Test cases will be marked "Red" when you copy/paste test case(s) across different local sessions.
- Test case execution marks the object whenever there is a change in the selected test case scope.
- "Check -in" or "Refresh" action will set the test case scope in the target view to default (Project selection) when the selected scope is not available in the target view. The target view can be Project server / Local session.

System Test

3

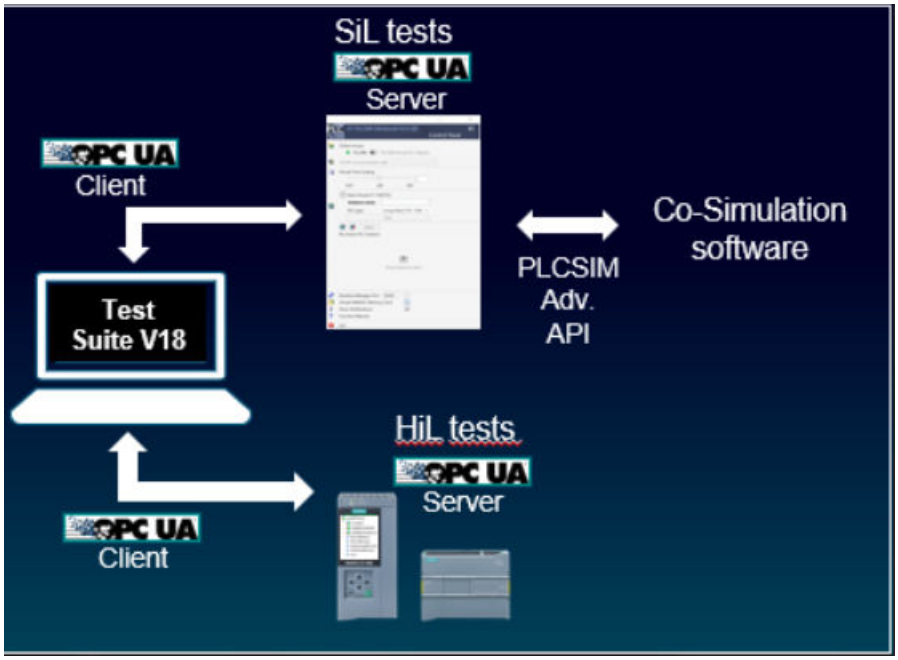
3.1 Introduction to System test

Introduction

System test enables the user to define and execute test cases for a PLC program by using OPC UA server interfaces.

System test allows to achieve the below tests

- Hardware in the loop tests with S7-1200 and S7-1500
- Software in the loop tests with PLCSIM Advanced.



System test allows you to define the test case using the below server interface files

- User defined server interface .XML file created in TIA portal.
- Standard SIMATIC server interface .XML file.
- Server interfaces created using Siemens OPC UA Modelling Editor (SiOME) tool

You can define the test case in the below pattern :

- Arrange : Set the values of the data that is passed to the program blocks under test.
- Wait: Wait for the specified time before verifying the action under test.
- Assert : Verify that the action under test behaves as expected.

You will run the tests with OPC UA server which is already in online state , hence make sure the PLC which is under test is acting as a OPC UA server and in the RUN mode for the successful execution of test cases defined.

Defining a Test Case

- Test case contains a variable definition section (optional)
- Test case will have multiple test steps to perform the following actions
  - Value assignments to server interface variables.
  - WAIT statement: Wait for the specified time without any action.
  - Assert statements: Compares actual values with the expected values.
- Execution of a test case will automatically perform the below mentioned steps without user interaction:
  - Performs precondition checks.
  - Connects to OPC UA server mentioned in the test case "Scope"
  - Executes a test case
  - Displays test results in the inspector window
  - Saves test results under common data\Logs
  - Disconnects from the OPC UA server

Benefits

- Support of Test-Driven Development.
- PLC code can be tested on real hardware.

3.2 Creating and Managing Test cases

Introduction

This chapter guides you to create and manage test cases. It also assists you in performing general test case operations.

Test case Editor

The test case editor allows you to edit each test case individually. It also enables you to perform the test case operations individually for each test case.

Prerequisites

- STEP 7 Professional V19 installed.
- Optional Package TIA Portal Test Suite Advanced V19 is installed.

Note

As an example, we will use "System test case_1" throughout the System test chapter. The user can use any name when creating a test case.

Adding a New Test case

You can define a test case which contains several rules against the PLC objects that can be validated. The test cases added are as explained below:

Procedure

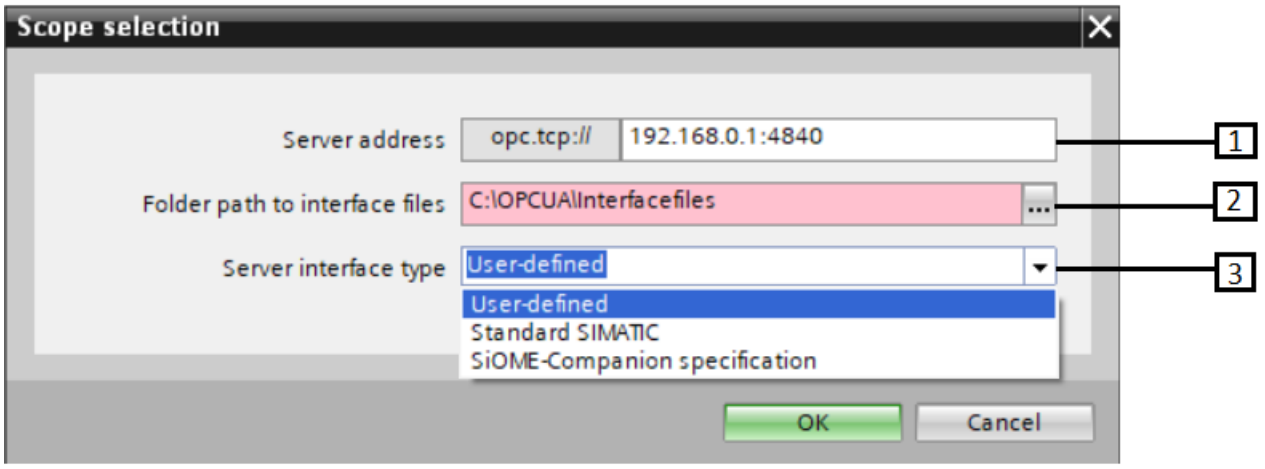
1. In the Test Suite project that you have created, double click "Add new test case". A new test case "System test case_1" is added to the project tree.

Note

- Test cases are sorted automatically after adding a new entry to the project tree.
- Test cases are sorted as per the precedence of unicode characters (Similar to project tree objects sorting).

How to set the scope of a test case ?

You can define the scope to execute the test case with the below parameters.


Scope selection
Server address
opc.tcp://
192.168.0.1:4840
Folder path to interface files
C:\OPCUA\Interfacefiles
...
Server interface type
User-defined
User-defined
Standard SIMATIC
SiOME-Companion specification
OK
Cancel
1
2
3

1. **OPC UA server address**
   You can specify the target server address to execute a test case in the following format.
   `opc.tcp://server:port/path`
   Examples:
   opc.tcp://192.168.0.1:4840
   opc.tcp:///localhost:51210/UA/SampleServer
   opc.tcp://opcua.demo-this.com:51210/UA/SampleServer
2. **Folder path to interface files**
   You can define the test cases by specifying the folder path to the interface files [User defined / standrad SIMATIC server interface/ SiOME - Companion specification]
   With the help of Intellisense, appropriate errors are displayed in case of any incorrect or unaccessible variables.
   Folder path will be shown as error, incases where
   - Path is invalid
   - Path is not available
   - Path doesn't exists

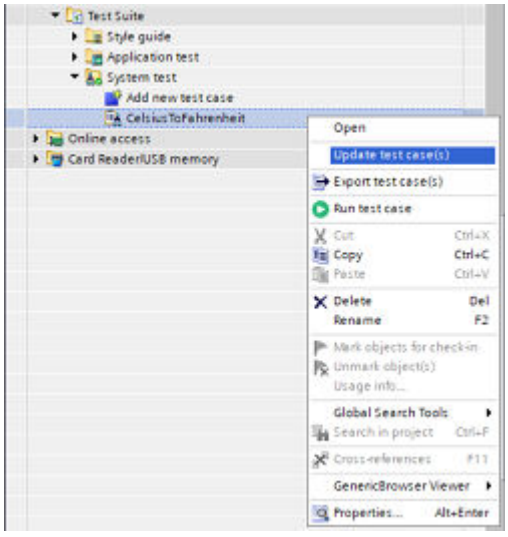
   Note
   - A warning message "Please select the path to server interface files" will be shown in "Syntax" tab until the folder path to interface files are specified.
   - Intellisense functionality will not be available to define the test case if you don't select the appropriate folder path to target server interface files.
   - Incorrect usage of variables cannot be realized until you specify the valid folder path to the target interface files
3. **Interface selection**
   You can select "User defined server interface " or "standard SIMATIC server interface" or "SiOME - Companion specification" based on your server configuration to define and execute the test case.
   - “User defined server interface” selection will help you to define the test cases for the offline configuration of the Step 7 program that is available in all the “User defined server interface” .XML files in the mentioned path.
   - “Standard SIMATIC server interface” selection will help you to define the test cases for the offline configuration of the Step 7 program that is available in the “standard SIMATIC server interface” .XML latest file in the mentioned path.
   - “SiOME - Companion specification interface” selection will help you to define the test cases for the offline configuration of the Step 7 program that is available in the “SiOME - Companion specification interface” .XML latest file in the mentioned path.

   You can find the details about the server interface files which are read successfully/partially in the “General” tab of the inspector window after mentioning the folder path at “Folder path to interface files” with the help of server interface you have selected.

Note

Update Test Case(s):

The test case content can be refreshed from context menu entry “Update test case(s)”. This option updates the test data layout from any changes in the target server interface files.



Deleting a Test case

You can delete a test case from the project tree and deleting will remove the same from the project tree under the "System test" subfolder.

Procedure

1. Under the "Project tree" area, select "Project 1 > Test Suite > System test > System test case_1".
2. Right click "System test case_1 > Delete".
3. The selected test case is now deleted.

Note

"Confirm deleting" dialog box will appear after you select test case(s) to be deleted. You can choose "Yes" or "No" based on your requirements.

Renaming a Test case

You can rename a test case in the project tree. After you rename test case(s), they will be sorted in an order of character precedence similar to the program blocks folder. You can rename test case(s) using the following methods:

Procedure

1. Under the "Project tree" area, select "Project 1 > Test Suite > System test > System test case_1".
2. Right click "System test case_1 > Rename".
3. Rename the test case according to your choice.

Note

- A test case name can accept a maximum of 125 unicode charaters
- When multiple test cases are selected, you can rename the last selected test case.
- All unicode characters are accepted except the following windows characters that are reserved
  - < (less than)
  - >(greater than)
  - : (colon)
  - " (double quote)
  - / (forward slash)
  - \ (backslash)
  - | (vertical bar or pipe)
  - ? (question mark)
  - * (asterisk)
  - white space

Opening a Test case Editor

You can open a test case editor and edit it separately by the following methods:

1. Double click on the required test case.
2. Select the required test case and click "Open"
3. Select the required test case and press "enter"
4. Select the required test case, pause and use mouse click.

Note

You also have the option to open multiple test cases.

Copy/Paste a Test case from project tree

You can copy test case(s) from the project tree. This results in additon of the added test case to the respective project tree under the "System test" subfolder with the name similar to project tree objects. It can be done in the following ways:

1. Press "Ctrl + C".
2. Click "Copy" from the context menu.
3. Click "Copy" from the main menu.
4. Tool bar icon

Note

- You can copy single/multiple test cases from the project tree.
- Test cases are sorted automatically after you perform the paste operation based on the precedence of the unicode char
- Copy/Paste operation of a test case works only witihn the same project, and not across different projects

3.3 Aliasing a server interface variable

Aliasing a server interface variable

System test enables the user to declare an alias (reference) variable to the server interface variable for the data types supported within the VAR/END_VAR section as mentioned below:

```
VAR
<alias_name> : <OPCUA_Server_Interface_Variable> ;
END_VAR
```

It is also possible to assign a value to the alias variables in the VAR section,

```
VAR
<alias_name> : <OPCUA_Server_Interface_Variable>  := <value> ;
END_VAR
```

Here “OPCUA_Server_Interface_Variable” can be accessed with the same hierarchy as it is represented in the server interface file

Example:

```
VAR
motorStart : "ServerInterface_DB".“InstDB".boolVar := TRUE;
"motor Start" : "ServerInterface_DB".“InstDB".boolVar := TRUE;
END_VAR
```

Where, `motorStart` is the alias name for the `boolVar` which is present in `“InstDB”` of selected server interface file.

Note

Test case variables must be in double quotation marks in case the name contains special characters or space and server interface variables must be always in double quotation marks as the User defined server interface editor in TIA portal allows all possible characters to define server nodes .

User has the access to provide alias for all the variables of the below mentioned Data blocks/PLC tags:

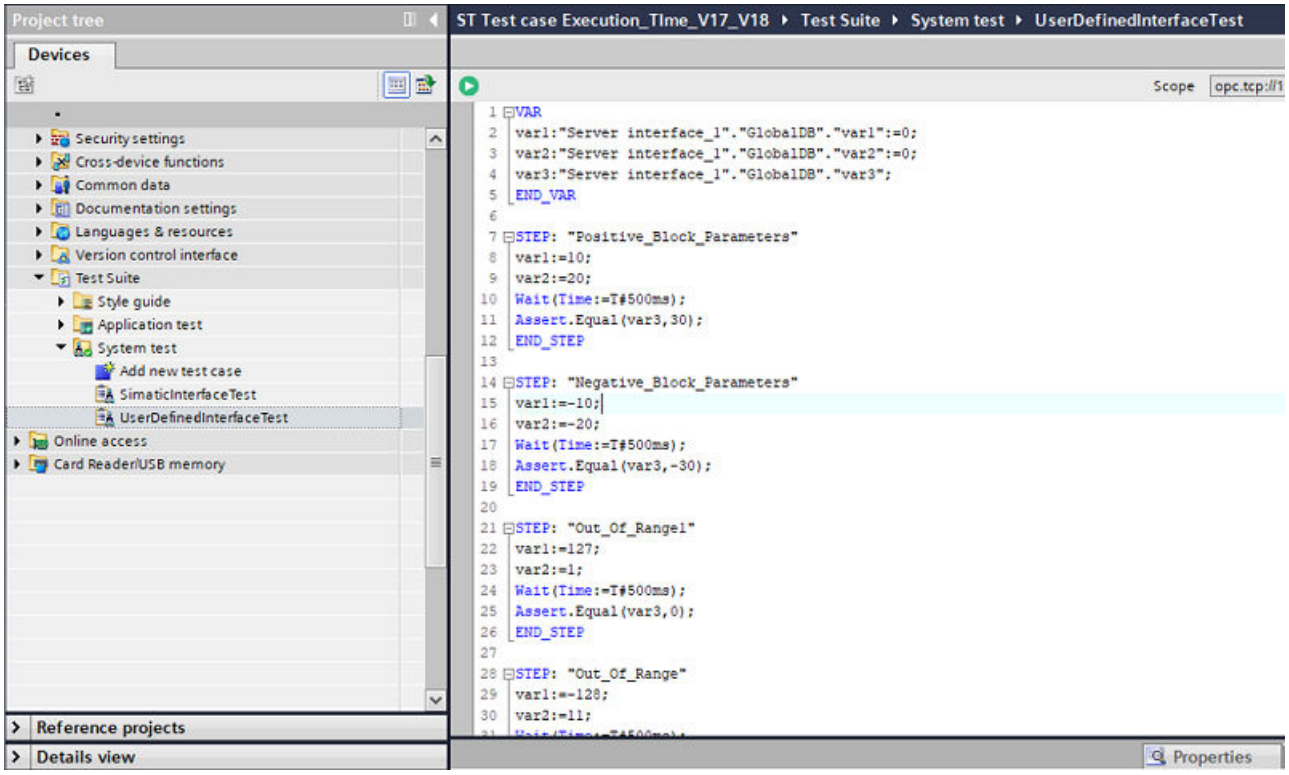
- Instance DB variables (Single/Multi-instance/Parameter instance)
- Global DB variables
- PLC tags

Examples

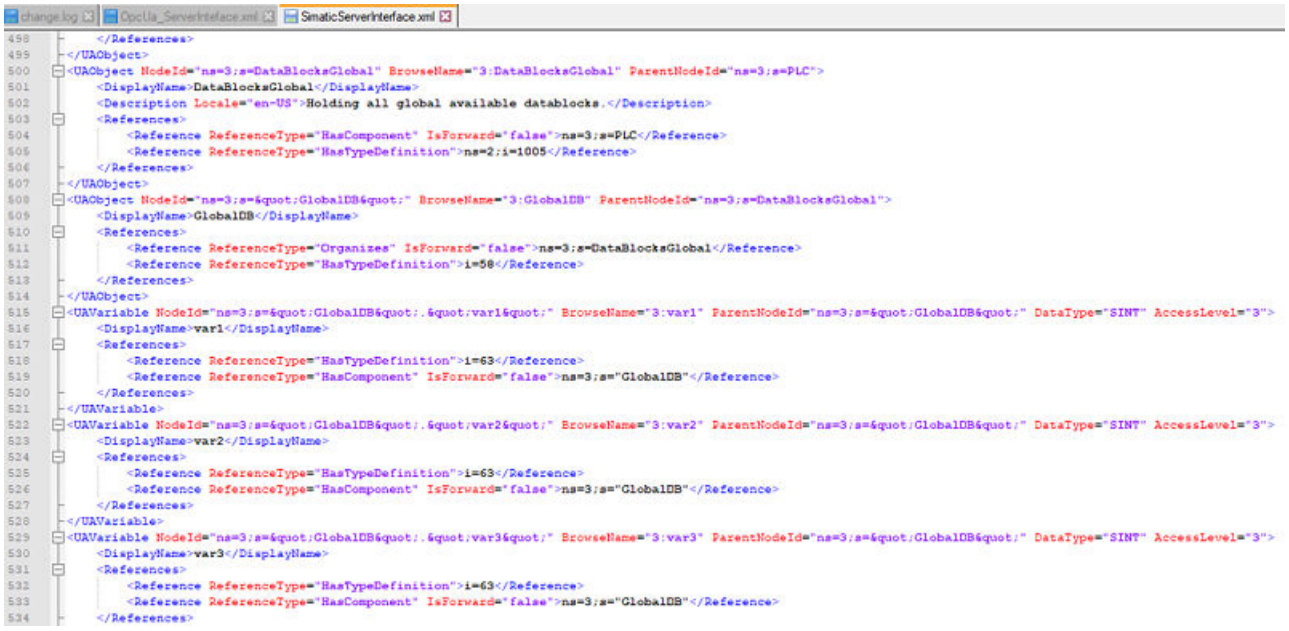
User defined server interface variables

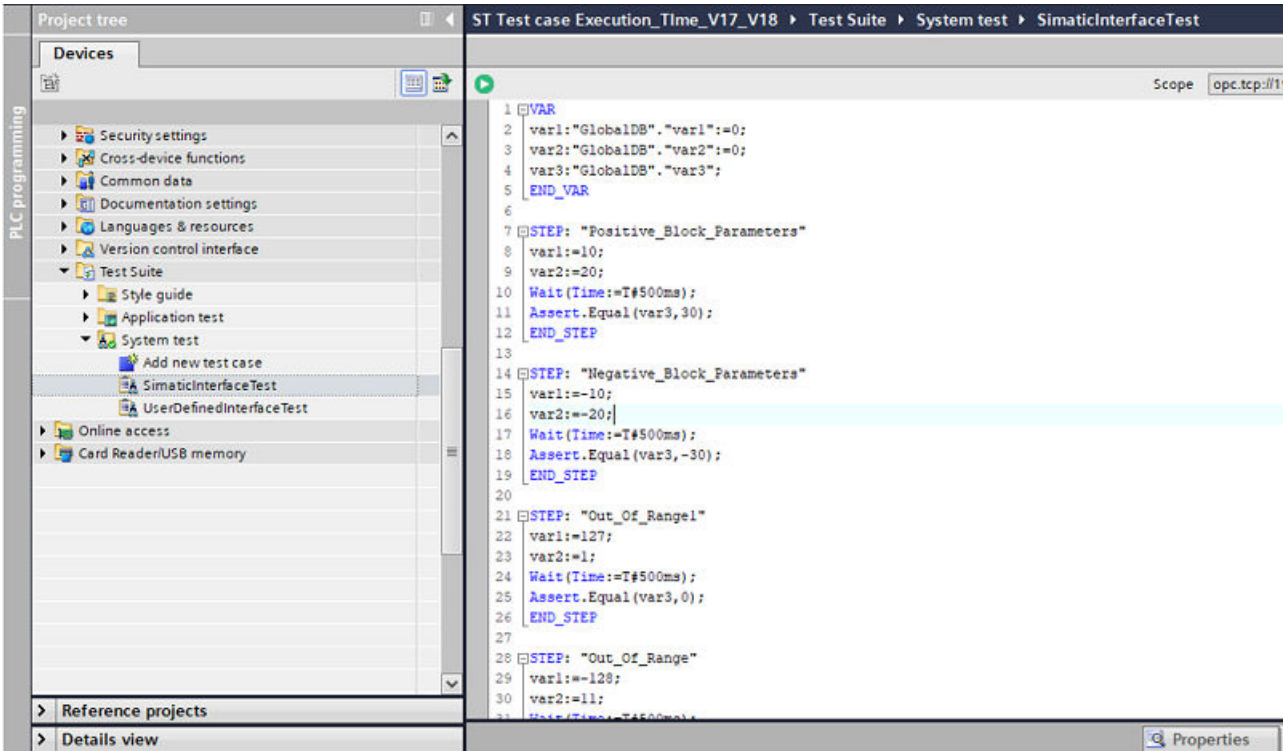
change.log | OpcUa_ServerInteface.xml

```
<UAObject NodeId="ns=2;i=2" BrowseName="2:GlobalDB">
  <DisplayName>GlobalDB</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=58</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=3</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=4</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=5</Reference>
  </References>
</UAObject>
<UAVariable NodeId="ns=2;i=3" BrowseName="2:var1" DataType="SINT" AccessLevel="3">
  <DisplayName>var1</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
  </References>
  <Extensions>
    <Extension>
      <si:VariableMapping>"GlobalDB"."var1"</si:VariableMapping>
    </Extension>
  </Extensions>
</UAVariable>
<UAVariable NodeId="ns=2;i=4" BrowseName="2:var2" DataType="SINT" AccessLevel="3">
  <DisplayName>var2</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
  </References>
  <Extensions>
    <Extension>
      <si:VariableMapping>"GlobalDB"."var2"</si:VariableMapping>
    </Extension>
  </Extensions>
</UAVariable>
<UAVariable NodeId="ns=2;i=5" BrowseName="2:var3" DataType="SINT" AccessLevel="3">
  <DisplayName>var3</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
  </References>
  <Extensions>
```



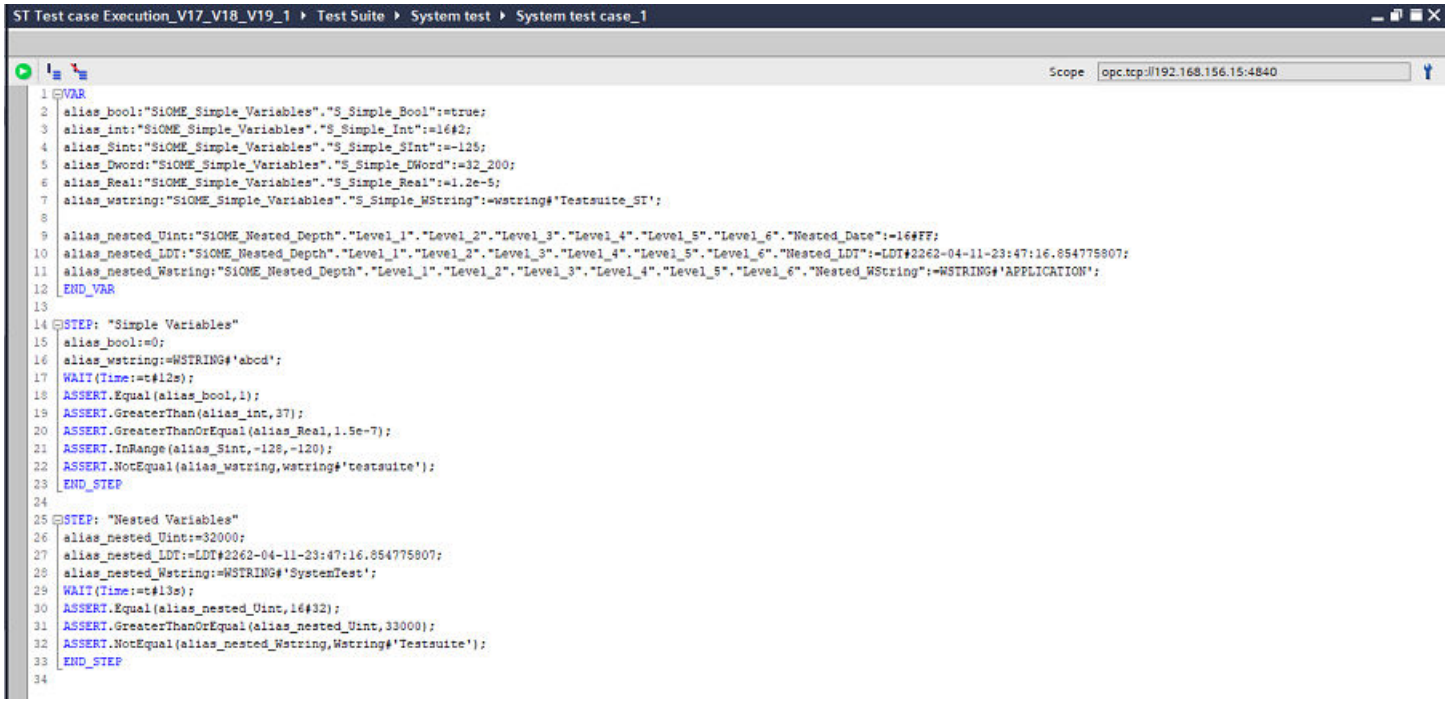
Standard SIMATIC server interface variables





SiOME - Companion Specification interface

```
<?xml version="1.0" encoding="utf-8"?>
<UANodeSet LastModified="2023-08-08T04:50:41.601Z" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://opcfoundation.or
    <NamespaceUris>
        <Uri>http://TestSuite_SiOME</Uri>
    </NamespaceUris>
    <Models>
        <Model ModelUri="http://TestSuite_SiOME" PublicationDate="2023-06-23T15:19:16+05:30" Version="1.00">
            <RequiredModel ModelUri="http://opcfoundation.org/UA/" PublicationDate="2021-09-15T00:00:00Z" Version="1.04.10"/>
        </Model>
    </Models>
    <Aliases>
        <Alias Alias="Boolean">i=1</Alias>
        <Alias Alias="SByte">i=2</Alias>
        <Alias Alias="Byte">i=3</Alias>
        <Alias Alias="Int16">i=4</Alias>
        <Alias Alias="UInt16">i=5</Alias>
        <Alias Alias="Int32">i=6</Alias>
        <Alias Alias="UInt32">i=7</Alias>
        <Alias Alias="Int64">i=8</Alias>
        <Alias Alias="UInt64">i=9</Alias>
        <Alias Alias="Float">i=10</Alias>
        <Alias Alias="Double">i=11</Alias>
        <Alias Alias="DateTime">i=13</Alias>
        <Alias Alias="String">i=12</Alias>
        <Alias Alias="HasComponent">i=47</Alias>
        <Alias Alias="HasProperty">i=46</Alias>
        <Alias Alias="Organizes">i=35</Alias>
        <Alias Alias="HasSubtype">i=45</Alias>
        <Alias Alias="HasTypeDefinition">i=40</Alias>
        <Alias Alias="HasModellingRule">i=37</Alias>
        <Alias Alias="HasEncoding">i=38</Alias>
    </Aliases>
    <Extensions>
        <Extension>
            <si:Generator Product="SiOME" Edition="Standard" Version="2.7.1"/>
        </Extension>
        <Extension>
            <si:GeneratorExtension Hash="2f7634509f67eac7cc56977d55797b3e"/>
        </Extension>
    </Extensions>
    <UAObject SymbolicName="http___TestSuite_SiOME" NodeId="ns=1;i=5000" BrowseName="1:http://TestSuite_SiOME" ParentNodeId="i=11715">
        <DisplayName>http://TestSuite_SiOME</DisplayName>
        <References>
            <Reference ReferenceType="HasComponent" IsForward="false">i=11715</Reference>
            <Reference ReferenceType="HasTypeDefinition">i=11616</Reference>
        </References>
    </UAObject>
    <UAVariable DataType="Boolean" NodeId="ns=1;i=6000" BrowseName="IsNamespaceSubset" ParentNodeId="ns=1;i=5000">
        <DisplayName>IsNamespaceSubset</DisplayName>
        <References>
            <Reference ReferenceType="HasProperty" IsForward="false">ns=1;i=5000</Reference>
            <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
        </References>
```



3.4 Basic Instructions

Basic Instructions Supported in Test Case Editor

A test case editor can have multiple test steps with the syntax as:

```
STEP : <step_name>
END_STEP
```

A step will have below mentioned statements:

Assignments

Assignment statements supported with the syntax as "`Var name`" := <value>; where "`Var name`" can be a server interface variable or an alias variable.

Aliasing a variable

The variables defined can be aliased to a new variable which can be subsequently used in the statements in place of the original variable.

Syntax: `<Operand1> : <Operand2> := <Value> ;`

where Operand1 is the alias variable of the original variable Operand 2 with a value assigned to it.

Example: `Cycles : "MyServerinterface"."ConveyorCycle"."Machine1" := 25;`

where Cycles is an alias variable of the original variable "MyServerinterface"."ConveyorCycle"."Machine1" with a value 25 assigned to it.

Please find more info on Aliasing a server interface variable (Page 76).

Wait Instruction

The Wait() instruction is used to wait for a specific time before reading the data from the connected OPC UA server.

Syntax: `WAIT(Time := <Value>);`

Example: `WAIT(Time := T#10ms);`

Parameters

The following table shows the parameters of the instruction:

Parameter	Data Type	Value	Description
Time	Time	Constant	Dedicated Time Span

Assert Statements

The ASSERT statements, compares the actual value of the server interface variable with the expected value specified by the user.
Syntax: `Assert.<_Condition_>(<actual_value>,<expected value>);`

Parameters

The following table shows the parameters of the instruction:

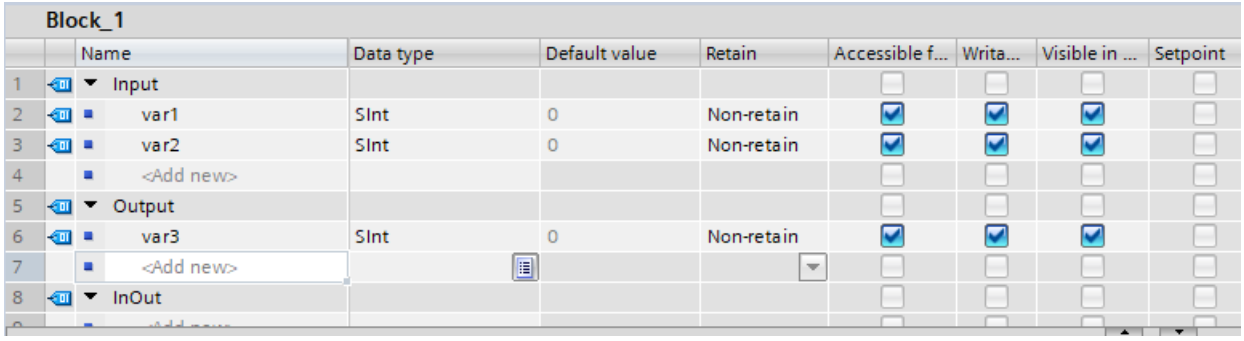
Instruction	Parameters	Description	Data type
Assert.Equal()	actual_value, expected_value	Assert.Equal validates whether the actual value is equal to expected value	Integers /Binary numbers /Floating point numbers/Character Strings/ Timers/ Date and Time
Assert.NotEqual()	actual_value, expected_value	Assert.NotEqual validates whether the first parameter is not equal to expected value	Integers /Binary numbers /Floating point numbers/Character Strings/ Timers/ Date and Time
Assert.GreaterThan()	actual_value, expected_value	Assert.GreaterThan validates whether the first parameter is greater than the expected value	Integers /Binary numbers /Floating point numbers/Character Strings /Timers/ Date and Time except BOOL datatype
Assert.GreaterThanOrEqual()	actual_value, expected_value	Assert.GreaterThanOrEqual validates whether the first parameter is greater than or equal to expected value	Integers /Binary numbers /Floating point numbers/Character Strings /Timers/ Date and Time except BOOL datatype
Assert.LessThan()	actual_value, expected_value	Assert.LessThan validates whether the first parameter is less than the expected value	Integers /Binary numbers /Floating point numbers/Character Strings /Timers/ Date and Time except BOOL datatype

Instruction	Parameters	Description	Data type
Assert.LessThanOrEqual()	actual_value, expected_value	Assert.LessThanOrEqual validates whether the first parameter is less than or equal to expected value	Integers /Binary numbers /Floating point numbers/Character Strings /Timers/ Date and Time except BOOL datatype
Assert.InRange()	actual_value, expected_min_value, expected_max_value	Assert.InRange helps to compare the expected value with actual value using minimum and maximum values defined	Integers /Floating point numbers

Note

For validating floating point numbers using Assert.InRange() is recommended.

Example:



Here the InstanceDB is Instance Data block of Block_1.

Test case for the above PLC program data:

```
VAR
var1:"Server interface_1"."InstanceDB"."var1":=0;
var2:"Server interface_1"."InstanceDB"."var2":=0;
var3:"Server interface_1"."InstanceDB"."var3";
END_VAR
```

```
STEP: "Positive_Block_Parameters"
var1:=10;
var2:=20;
Wait(Time:=T#500ms);
Assert.Equal(var3,30);
END_STEP
```

```
STEP: "Negative_Block_Parameters"
var1:=-10;
var2:=-20;
Wait(Time:=T#500ms);
Assert.Equal(var3,-30);
END_STEP
```

```
STEP: "Out_Of_Range1"
var1:=127;
var2:=1;
Wait(Time:=T#500ms);
Assert.Equal(var3,0);
END_STEP
```

```
STEP: "Out_Of_Range"
var1:=-128;
var2:=11;
Wait(Time:=T#500ms);
Assert.Equal(var3,0);
END_STEP
```

3.5 Supported Data Types

3.5.1 Binary Numbers

Binary Numbers

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Bool	Boolean	FALSE or TRUE	FALSE or TRUE	FALSE or TRUE
	Unsigned integers (decimal system)	0 or 1	0 or 1	
	Binary numbers	2#0 or 2#1	2#0 or 2#1	
	Hexadecimal numbers	16#0 or 16#1	16#0 or 16#1	

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Byte	Integers (decimal system)	Signed integers: -128 to +127 Unsigned integers: 0 to 255	15	16#0 to 16#FF
	Binary numbers	2#0 to 2#1111_1111	2#00001111	
	Hexadecimal numbers	16#0 to 16#FF	16#0F	
Word	Integers (decimal system)	Signed integers: -32_768 to +32_767 Unsigned integers: 0 to 65_535	61680	16#0 to 16#FFFF
	Binary numbers	2#0 to 2#1111_1111_1111_1111	2#1111000011110000	
	Hexadecimal numbers	16#0 to 16#FFFF	16#F0F0	
Dword	Signed integers (decimal system) Unsigned integers (decimal system)	-2_147_483_647 to +2_147_483_647. 0 to 4_294_967_295	0	16#0000_0000 to 16#FFFF_FFFF
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111	2#00000000111100001111111100001111	
	Hexadecimal numbers	16#0000_0000 to 16#FFFF_FFFF	16#00F0FF0F	
Lword	Integers (decimal system)	Signed integers: -9_223_372_036_854_775_808 to +9_223_372_036_854_775_807 Unsigned integers: 0 to 18_446_744_073_709_551_615	+26_123_590_360_715	16#0000_0000 to 16#FFFF_FFFF_FFFF_FFFF
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111	2#0000_0000_0000_0000_0000_1011_1110_0001_0010_1111_0101_0010_1101_1110_1000_1011	
	Hexadecimal numbers	16#0000_0000 to 16#FFFF_FFFF_FFFF_FFFF	16#0000_0000_5F52_DE8B	

3.5.2 Integers

Integers

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Sint	Signed integers (decimal system)	-128 to +127	44	-128 to +127
	Binary numbers (only positive)	2#0 to 2#0111_1111	2#0010_1100	
	Hexadecimal numbers (only positive)	16#0 to 16#7F	16#2C	
Int	Signed integers (decimal system)	-32_768 to +32_767	3_785	-32_768 to +32_767
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111	2#0000_1110_1100_1001	
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF	16#0EC9	
Dint	Signed integers (decimal system)	-2_147_483_648 to +2_147_483_647	+125_790	-2_147_483_648 to +2_147_483_647
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111_1111_1111_1111_1111	2#0000_0000_0000_0001_1110_1011_0101_1110	
	Hexadecimal numbers	16#0 to 16#7FFF_FFFF	16#0001_EB5E	
Usint	Unsigned integers (decimal system)	0 to 255	78	0 to 255
	Binary numbers	2#0 to 2#1111_1111	2#0100_1110	
	Hexadecimal numbers	16#0 to 16#FF	16#4E	
Uint	Unsigned integers (decimal system)	0 to 65_535	65_295	0 to 65_535
	Binary numbers	2#0 to 2#1111_1111_1111_1111	2#1111_1111_0000_1111	
	Hexadecimal numbers	16#0 to 16#FFFF	16#FF0F	
Udint	Unsigned integers (decimal system)	0 to 4_294_967_295	4_042_322_160	0 to 4_294_967_295
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111	2#1111_0000_1111_0000_1111_0000_1111_0000	
	Hexadecimal numbers	16#0 to 16#FFFF_FFFF	16#F0F0_F0F0	

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Lint	Signed integers (decimal system)	-9_223_372_036_854_775_808 to +9_223_372_036_854_775_807	+154_325_790_816_159	-9_223_372_036_854_775_808 to +9_223_372_036_854_775_807
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_1111_1111_1111_1111_1111_ 1111_1111_1111_1111_1111_1111_1111_1111	2#0000_0000_0000_0000_1000_1100_0101_1011_1100 _0101_1111_0000_1111_0111_1001_1111	
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF_FFFF_FFFF_FFFF	16#0000_8C5B_C5F0_F79F	
Ulint	Unsigned integers (decimal system)	0 to 18_446_744_073_709_551_615	154_325_790_816_159	0 to 18_446_744_073_709_551_615
	Binary numbers	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111 _1111_1111_1111_1111_1111_1111_1111_1111	2#0000_0000_0000_0000_1000_1100_0101_1011_1100 _0101_1111_0000_1111_0111_1001_1111	
	Hexadecimal numbers	16#0 to 16#FFFF_FFFF_FFFF_FFFF	16#0000_8C5B_C5F0_F79F	

3.5.3 Floating Point Numbers

Floating Point Numbers

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Real	Floating-point numbers according to IEEE754	-3.402823e+38 to -1.175495e-38	1.0e-5;	Only one type format supported as mentioned in value range
		±0.0		
	Floating-point numbers	+1.175495e-38 to +3.402823e+38	1.0;	

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Lreal	Floating-point numbers according to IEEE754	-1.7976931348623157 e+308 to -2.2250738585072014 e-308	1.0e-5;	Only one type format supported as mentioned in value range
		±0.0		
	Floating-point numbers	+2.225073858507201 4e-308 to +1.7976931348623157 e+308	1.0;	

3.5.4 Timers

Timers

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Time	Signed duration	T#-24d_20h_31m_23s_ 648ms to T#+24d_20h_31m_23s _647ms	T#10d_20h_30m_20s_ 630ms, TIME#10d_20h_30m_2 0s_630ms	T#-24d_20h_31m_ 23s_648ms to T#+24d_20h_31m _23s_647ms
LTIME	Signed duration	LT#-106751d_23h_47m _16s_854ms_775us_80 8ns to LT#+106751d_23h_47 m_16s_854ms_775us_ 807ns	LT#11350d_20h_25m_ 14s_830ms_652us_315 ns, LTIME#11350d_20h_25 m_14s_830ms_652us_ 315ns	LT#-106751d_23h _47m_16s_854ms _775us_808ns to LT#+106751d_23h _47m_16s_854ms _775us_807ns
S5Time	S7 time in increments of 10 ms (default value)	S5T#0MS to S5T#2H_46M_30S_0MS	S5T#10s, S5TIME#10s	S5T#10s

3.5.5 Strings

Strings

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
Char	CHAR	ASCII character set ASCII Values (32 - 127)	'A'(Characters which represents the ASCII value 32-127)	'Character e.g. 'A'
WChar	Unicode,ASCII Values	$0000 - $D7FF	'WCHAR#'a'	
String	ASCII character string incl. special characters	0 to 254 characters	'Name' STRING#'NAME' STRING#'Na... (The actual length of the string is longer than the space available on the screen.) STRING#'' (The string is empty.)	STRING#'NAME'
WString	Unicode character string; n specifies the length of the character string.	Preset value: 0 to 254 characters Max. possible value: 0 to 16382	'WSTRING#'Hello World' WSTRING#'Hello Wo... (The actual length of the string is longer than the space available on the screen.) WSTRING#'' (The string is empty.)	WSTRING#'Hello World'

3.5.6 Date and Time

Date and time

Data types	Data type formats	Value range	Examples	Value format displayed in test results
DATE	IEC date (Year-Month-Day)	D#1990-01-01 to D#2169-06-06	'D#2009-12-31, DATE#2009-12-31	D#1990-01-01 to D#2169-06-06
TIME_OF_DAY (TOD)	Time-of-day (hours:minutes:seconds.milliseconds)	TOD#00:00:00.000 to TOD#23:59:59.999	'TOD#10:20:30.400, TIME_OF_DAY#10:20:30.400	'TOD#23:59:59.999
LTOD (LTIME_OF_DAY)	Time-of-day (hours:minutes:seconds.nanoseconds)	LTOD#00:00:00.000000000 to LTOD#23:59:59.999999999	'LTOD#10:20:30.400_365_215, LTIME_OF_DAY#10:20:30.400_365_215	'LTOD#10:20:30.400_365_215

Data types	Data type formats	Value range	Examples	Value format displayed in test results
DATE_AND_TIME (date and time of day)	Date and time (year-month-day-hour:minute:second:millisecond))	Min.: DT#1990-01-01-00:00:00.000 Max.: DT#2089-12-31-23:59:59.999	'DT#2008-10-25-08:12:34.567, DATE_AND_TIME#2008-10-25-08:12:34.567	'DT#2008-10-25-08:12:34.567
LDT (DATE_AND_LTIME)	Date and time (Year-Month-Day-Hour:Minute:Second.Nanoseconds)	Min.: LDT#1970-01-01-00:00:00.000000000 Max.: LDT#2262-04-11-23:47:16.854775807	'LDT#2008-10-25-08:12:34.567	'LDT#2008-10-25-08:12:34.567
DTL	Date and time (Year-Month-Day-Hour:Minute:Second.Nanoseconds)	Min.: DTL#1970-01-01-00:00:00.0 Max.: DTL#2262-04-11-23:47:16.854775807	'Max.: DTL#2262-04-11-23:47:16.854775807 DTL#2008-12-16-20:30:20.250	'Character e.g. 'A'

Note

OPC UA server limits the support of nanoseconds precision for LTOD (LTIME_OF_DAY).

3.5.7 SiOME Companion specification datatypes

To write the tests for **"SiOME-Companion specification"** server interface, the following data types are supported.

OPCUA data type	Mapped SIMATIC data type
Boolean	BOOL
BYTE -> Byte	BYTE
	USINT
	CHAR
UINT -> UInt16	Word
	UINT
	S5TIME
	DATE
	WCHAR
UDINT -> UInt32	DWORD
	UDINT
	TIME_OF_DAY(TOD)
ULINT -> UInt64	LWORD
	ULINT
	LTIME_OF_DAY(LTOD)

OPCUA data type	Mapped SIMATIC data type
SByte	SINT
Int16	INT
DINT -> Int32	DINT
	TIME
LINT -> Int64	LINT
	LTIME
Float	REAL
Double	LREAL
DateTime	LDT
WSTRING	WSTRING
	STRING
Enumeration	DINT

Displayed Data type in the Test case editor intellisense

- Multiple mapped SIMATIC data types: Left side of "->" indicates the displayed Data type.
- Single data type mapping : Mentioned SIMATIC data type will be displayed

You can also find mapping of the respective OPC UA data type to SIMATIC data type in the below location
https://support.industry.siemens.com/cs/mdm/109798671?c=143871134091&lc=en-WW

3.6 Test Case Execution

Introduction

Test case execution can be done on the selected OPC UA server mentioned in the test case scope.You will be able to run the test cases with any SIMATIC PLC1200/PLC 1500 which is capable of acting as a OPC UA server.

Preconditions

The following preconditions should be met successfully before proceeding for the execution of test cases.

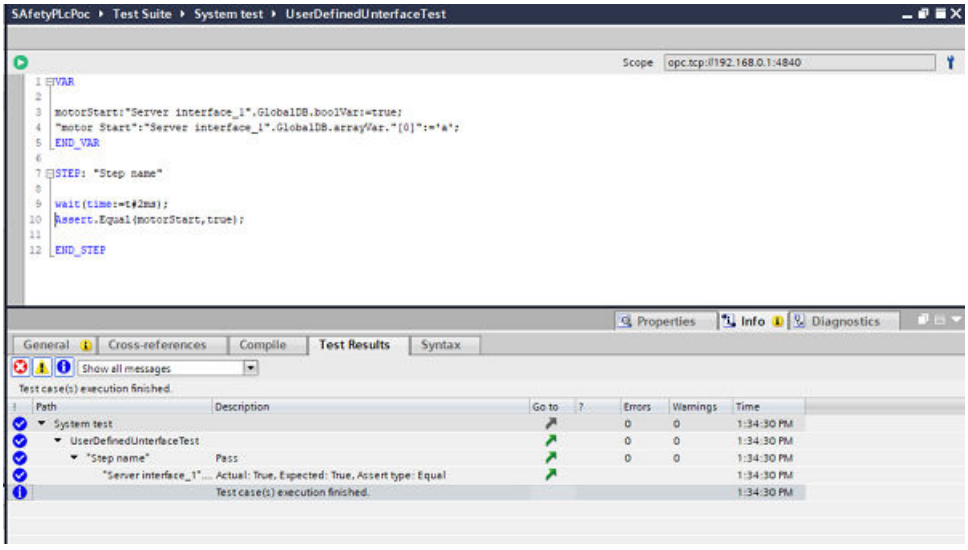
- Valid Test Suite license is available.
- Test case definition should be syntactically correct.
- Verify whether all variables used in the test case are error free. This can be performed only when the valid folder path is mentioned for the interface files.
- Valid OPC UA server address is mentioned in test case scope.
- Verify whether the server is of CPU type.

Execution

The test execution can be started in two different ways, either from the current opened test case editor or with a context menu entry on the test case in the project tree.

Execution of a test case(s) will include the following steps:

- Verifies, target OPC UA server should be of type PLC.
- Verifies whether the server/PLC is online and in RUN mode
- Connects to the PLC which is in RUN mode as Anonymous user.
- If the server is not of type PLC/ server is not online /PLC is not in RUN mode then the test case execution will be aborted.
- If the connection is successful then the test case execution will be continued by reading/ writing the data from/to the OPC UA server nodes. After the successful test case execution, connection with OPC UA server will be closed.
- Results will be displayed in the "Test Results" tab in the inspector window with 'Pass'/'Fail' feedback for each test step and the 'Go to" option will be provided to navigate to the specific test step.
- Log file will be created for each test case execution under common data\Logs.



Note

- You will be able to run the test cases with SIMATIC S7-1500 /S7-1200 which acts as an OPC-UA server
- During test case execution, the user will not be able to use TIA Portal.
- You can cancel the test case execution at any point of time. But the cancellation status Is displayed only after the specified waiting time, incase the waiting time is already started when the user cancelled the execution of a test case.
- Working with Safety blocks, It is expected that the PLC should be in RUN mode for at least one safety cycle before writing values to F-Inputs.
- System test supports OPC UA companion specification created using SiOME.

Note

Points to note while using SiOME-Companion Specification

- You can also use the server interfaces which defines instances and type definitions in different server interfaces / different namespaces as an input to write the test case.
- You will not be able to write the test for structure elements which are not defined as child nodes in information model.
- OPC UA variables of type arrays are not supported.
- If you have an information model with multiple OPC UA nodes with the same name but different node ID then the first occurred node in the selected folder path / OPC UA server interface connected will be considered to define/run the test case.

3.7 Test Case Execution Failure

This section helps you in evaluating reasons for test case execution failure.

If the following preconditions are not met, then you will not be able to run the test cases successfully:

- Valid OPC UA server address should be available
- Test case editor should be error free.
- Connection to target OPC UA server should be successful.
- Target PLC which is configured as OPC UA server address should be in Online and RUN mode
- Target server interface(s) should be the same as exported in server interface file(s) which are used to define the test case(s).
  - This helps in avoiding test case failures during execution.
  - This helps to maintain the test cases up to date with the target PLC program under test.

Error codes during test case execution

The following table comprises of error codes for connection errors.

Error ID	Description
OpcUa_BadCommunicationError	Connection timeout. **Possible causes:** The server address (ServerEndpointURL) is incorrect or incomplete. The server does not accept the buffer size sent by the client during connection establishment. **Remedies:** Check and correct the ServerEndpointURL. Check the permitted buffer size. The client of the S7-1500 CPU provides 8192 bytes. The server must accept this value.
OpcUa_BadTimeout	A network timeout occurred. **Possible causes:** Connection to the OPC UA server is too slow. The network load is too high. The OPC UA server is not available. The server might not accept the buffer size sent by the client during connection establishment. **Remedies:** Check the URL of the OPC UA server. Increase the timeout value of the instruction OPC_UA_Connect. Check the permitted buffer size. The client of the S7-1500 CPU provides 8192 bytes. The server must support this value.
OpcUa_BadNotSupported	**Possible causes:** The server address (ServerEndpointURL) is incorrect or incomplete. Server does not support the requested operation.
OpcUA_BadServerHalted	The server has stopped and cannot process any requests
OpcUA_BadUserAccessDenied	User does not have permission to perform the requested operation.
OpcUA_BadSessionIdInvalid	The session id is not valid.
OpcUA_BadSessionClosed	The session was closed by the client.
OpcUA_BadSessionNotActivated	The session cannot be used because ActivateSession has not been called.
OpcUA_BadServerUriInvalid	The ServerUri is not a valid URI.
OpcUa_BadSecurityModeRejected	The security mode does not meet the requirements set by the Server. Server is setting higher security requirements. **Remedy:** Use a higher security setting for the connection to the server.
OpcUa_BadSecurityPolicyRejected	The security policy does not meet the requirements set by the Server.
OpcUa_BadTooManySessions	The server has reached the maximum number of sessions.
OpcUa_BadNoValidCertificates	The client did not provide at least one software certificate that is valid and meets the profile requirements for the server.
OpcUa_BadTcpNotEnoughResources	The client's connection resources required to establish the connection are used up. Possible remedy: If the connection is established and terminated in rapid succession, the frequency must be reduced.
OpcUa_BadConnectionRejected	Could not establish a network connection to remote server.
OpcUa_BadDisconnect	The server has disconnected from the client.
OpcUa_BadConnectionClosed	The network connection has been closed. The connection/session could not be ""reactivated"" automatically. Possible cause: Session deleted on the server, e.g. due to restart or timeout.
TestSuite_BadNotSupportedDeviceType	The target PLC device is not supported.
TestSuite_BadDeviceNotInRunState	The target PLC is not in run state.

The following table comprises of error codes for failure of assignment operation.

Error ID	Description
OpcUa_BadInternalError	An internal error occurred as a result of a programming or configuration error. **Possible cause:** You have write-accessed a single element of a tag of data type DTL. Write access to tags of data type DTL is only possible to the complete structure.
OpcUa_BadViewIdUnknown	The view id does not refer to a valid view Node.
OpcUa_BadViewTimestampInvalid	The view timestamp is not available or not supported.
OpcUa_BadViewParameterMis-matchInvalid	The view parameters are not consistent with each other.
OpcUa_BadViewVersionInvalid	The view version is not available or not supported.
OpcUa_BadNothingToDo	There was nothing to do because the Client passed a list of operations with no elements.
OpcUa_BadTooManyOperations	The request could not be processed because it specified too many operations.
OpcUa_BadNodeIdInvalid	The syntax of the node id is not valid.
OpcUa_BadNodeIdUnknown	The node id refers to a node that does not exist in the server address space.
OpcUa_BadAttributeIdInvalid	The attribute is not supported for the specified Node.
OpcUa_BadNotFound	A requested item was not found or a search operation ended without success.
OpcUa_BadNotReadable	The access level does not allow reading the Node.
OpcUa_BadNotSupported	The requested operation is not supported.
OpcUa_BadSourceNodeIdInvalid	The source node id does not refer to a valid node.
OpcUa_BadStructureMissing	A mandatory structured parameter was missing or null.
OpcUa_BadServerNotConnected	The operation could not complete because the client is not connected to the server.
OpcUa_BadServerHalted	The server has stopped and cannot process any requests.
OpcUa_BadDataTypeIdUnknown	The extension object cannot be (de)serialized because the data type id is not recognized.
OpcUa_BadUserAccessDenied	User does not have permission to perform the requested operation.
OpcUa_BadBrowseDirectionInvalid	The browse direction is not valid.
OpcUa_BadParentNodeIdInvalid	The parent node id does not to refer to a valid node.
OpcUa_BadNodeIdRejected	The requested node id was reject because it was either invalid or server does not allow node ids to be specified by the client.
OpcUa_BadNodeIdExists	The requested node id is already used by another node
OpcUa_BadNodeClassInvalid	The node class is not valid.
OpcUa_BadBrowseNameInvalid	The browse name is invalid.
OpcUa_BadBrowseNameDuplicated	The browse name is not unique among nodes that share the same relationship with the parent
OpcUa_BadNodeAttributesInvalid	The node attributes are not valid for the node class.
OpcUa_BadTypeDefinitionInvalid	The type definition node id does not reference an appropriate type node.
OpcUa_BadTypeMismatch	The value supplied for the attribute is not of the same type as the attribute's value.
OpcUa_BadTcpServerTooBusy	The server cannot process the request because it is too busy.
OpcUa_BadDeviceFailure	There has been a failure in the device/data source that generates the value that has affected the value.
OpcUa_BadSensorFailure	There has been a failure in the sensor from which the value is derived by the device/data source.

Error ID	Description
TestSuite_BadDatatypeNotSupported	The datatype is not supported
TestSuite_BadNotSupported	The selected test case scope is invalid. Hint - standard SIMATIC server interface is not available for PLC1200 controller
TestSuite_BadOutsideValidValueRange	The value is outside the valid value range of the datatype

The following table comprises of the error codes for failure of assert operation.

Error	Description
OpcUa_BadViewIdUnknown	The view id does not refer to a valid view Node.
OpcUa_BadViewTimestampInvalid	The view timestamp is not available or not supported.
OpcUa_BadViewParameterMismatchInvalid	The view parameters are not consistent with each other.
OpcUa_BadViewVersionInvalid	The view version is not available or not supported.
OpcUa_BadNothingToDo	There was nothing to do because the Client passed a list of operations with no elements.
OpcUa_BadTooManyOperations	The request could not be processed because it specified too many operations.
OpcUa_BadNodeIdInvalid	The syntax of the node id is not valid.
OpcUa_BadNodeIdUnknown	The node id refers to a node that does not exist in the server address space.
OpcUa_BadAttributeIdInvalid	The attribute is not supported for the specified Node.
OpcUa_BadNotFound	A requested item was not found or a search operation ended without success.
OpcUa_BadNotReadable	The access level does not allow reading the Node.
OpcUa_BadNotSupported	The requested operation is not supported.
OpcUa_BadNotWritable	The access level does not allow writing to the Node.
OpcUa_BadOutOfRange	The value was out of range.
OpcUa_BadSourceNodeIdInvalid	The source node id does not refer to a valid node.
OpcUa_BadStructureMissing	A mandatory structured parameter was missing or null.
OpcUa_BadServerNotConnected	The operation could not complete because the client is not connected to the server.
OpcUa_BadServerHalted	The server has stopped and cannot process any requests.
OpcUa_BadDataTypeIdUnknown	The extension object cannot be (de)serialized because the data type id is not recognized.
OpcUa_BadUserAccessDenied	User does not have permission to perform the requested operation.
OpcUa_BadRequestHeaderInvalid	The header for the request is missing or invalid.
OpcUa_BadStructureMissing	A mandatory structured parameter was missing or null.
OpcUa_BadContinuation PointInvalid	The continuation point provide is longer valid.
OpcUa_BadNoContinuationPoints	The operation could not be processed because all continuation points have been allocated.
OpcUa_BadBrowseDirectionInvalid	The browse direction is not valid.

Error	Description
OpcUa_BadParentNodeIdInvalid	The parent node id does not to refer to a valid node.
OpcUa_BadNodeIdRejected	The requested node id was reject because it was either invalid or server does not allow node ids to be specified by the client.
OpcUa_BadNodeIdExists	The requested node id is already used by another node
OpcUa_BadNodeClassInvalid	The node class is not valid.
OpcUa_BadBrowseNameInvalid	The browse name is invalid.
OpcUa_BadBrowseNameDuplicated	The browse name is not unique among nodes that share the same relationship with the parent
OpcUa_BadNodeAttributesInvalid	The node attributes are not valid for the node class.
OpcUa_BadTypeDefinitionInvalid	The type definition node id does not reference an appropriate type node.
OpcUa_BadTypeMismatch	The value supplied for the attribute is not of the same type as the attribute's value.
OpcUa_BadTcpServerTooBusy	The server cannot process the request because it is too busy.
OpcUa_BadDeviceFailure	There has been a failure in the device/data source that generates the value that has affected the value.
OpcUa_BadSensorFailure	There has been a failure in the sensor from which the value is derived by the device/data source.
TestSuite_BadInvalidOperation	The comparision type used in assert statement is not valid
TestSuite_BadTypeExpValue1MisMatch / TestSuite_BadTypeExpValue2MisMatch	The value used in assert statement is outside the valid value range supported by the data type
TestSuite_BadValueNull	Requested Non-Nullable item is null.

3.8 Additional Features

Introduction

System test supports the following features:

1. Master copy support
2. Import/Export of test cases

Master copy support

Master copy support is used to transmit system tests to other projects. It is possible to create a master copy out of an system test case. Master copies are also used to create standardized copies of frequently used system test cases.You can create as many items as needed and then insert them into the project.

You can store master copies either in the project library or in a global library. Master copies in the project library can only be used within the project. When you create a master copy in a global library, it can be used in different projects.

Import/Export of test cases

You can import and export the test case(s) added in the editor to an external textual file with the following options:

- **Export**: Context menu entry "Export test case(s)" in project tree when you select System test folder/test case(s).
- **Import**: Context menu entry "Import test case(s)" in project tree when you select System test folder.

Note

System test exports the test cases with .tst file extension. It is also possible to import the .tat files into System test successfully without any scope information.

Sample of .tst textual file:

```
TEST_CASE "SingleStep"

PROPERTY
AUTHOR : "Tom"
VERSION : "0.1"
COMMENT : "This testcase contains single step"
SERVER_ADDRESS : "opc.tcp://192.168.0.1:6"
FOLDER_PATH : "D:\Simatic"
// The selected server interface can be User defined, Standard
SIMATIC or SiOME-Companion specification
SELECTED_SERVER_INTERFACE : "Standard_SIMATIC"
END_PROPERTY

VAR

tcvar1 : fbMotor_DB.outputs.myVar1:=1;
tcvar2 : fbMotor_DB.outputs.myVar2:=1;
tcvar3 : fbMotor_DB.outputs.myVar3:=1;

END_VAR

STEP: "Step name"

wait(time:=t#2ms);
Assert.Equal(tcvar1,1);
Assert.Equal(tcvar2,1);
Assert.Equal(tcvar3,1);
END_STEP

END_TEST_CASE
```

Errors and Warnings messages for import operation:

You will also be shown status information, warnings and error messages in the Inspector window under the "Info > General" tab.

Below are the different feedback messages provided for import operation:

- Import of test case(s) completed with warnings, when:
  - The test case property 'Author' imported with the value which is not valid (Example: more than 125 characters/only space allowed).
  - The test case property 'Version' imported with the value which is not valid (Example: other than range 0.0 to 99.99).
  - The test case scope imported as NULL when the scope in the file is not available in the project.
- Import of test case(s) completed with errors, when:
  - TEST_CASE keyword is missing in the textual file.
  - Test case name is missing in the textual file.
  - The test case name not enclosed in double quotes in textual file.
  - The test case name contains the characters which are not allowed (Example: <, >, :, ", /, \, |, ?, *, space) in textual file.
  - The test case name contains only NULL/Empty string in textual file.
  - The END_TEST_CASE keyword missing in textual file.

When you abort the import operation then the test case which is already in progress to import will be terminated immediately.

Multiuser engineering

System test supports modification and execution of rule sets in Multiuser engineering environment.

Note

- System test case(s) will be marked "Red" when you copy/paste system test case(s) across different local sessions.
- "Check -in" or "Refresh" action will set the folder path to interface files in the target view to default (Project selection) when the selected folder path to interface files is not available in the target view. The target view can be Project server / Local session.

Refer TIA Portal Multiuser engineering section for more information.

Openness Support

4

4.1 Introduction

Overview

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

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

Connecting to the TIA Portal

TIA Portal Test Suite is an optional package on TIA Portal to increase the efficiency of the PLC programmer by ensuring high quality. To use Test Suite services in the Openness application it is must to install the Test Suite package.

Available namespaces for Test Suite objects

The following namespaces are available for the Test Suite interfaces for features Style guide and Application test:

- `Siemens.Engineering.TestSuite`
- `Siemens.Engineering.TestSuite.StyleGuide`
- `Siemens.Engineering.TestSuite.ApplicationTest`
- `Siemens.Engineering.TestSuite.SystemTest`
- `Siemens.Engineering.TestSuite.StyleGuide.RuleSetExecutor`
- `Siemens.Engineering.TestSuite.ApplicationTest.TestCaseExecutor`
- `Siemens.Engineering.TestSuite.SystemTest.TestCaseExecutor`
- `Siemens.Engineering.TestSuite.TestResults`

Accessing Test Suite as a service

Application

You can access the Test Suite service and its methods in Openness application on the target TIA Portal project using the below code.

Program Code

To access Test Suite as a service:

```
Using Siemens.Engineering;
Using Siemens.Engineering.TestSuite;

TestSuiteService testSuite = project.GetService<TestSuiteService>( ) ;
```

Here 'project' is an instance of the TIA Portal project retrieved in the openness application.

Note

The above program code will give you an build error if the Test Suite package is not installed.

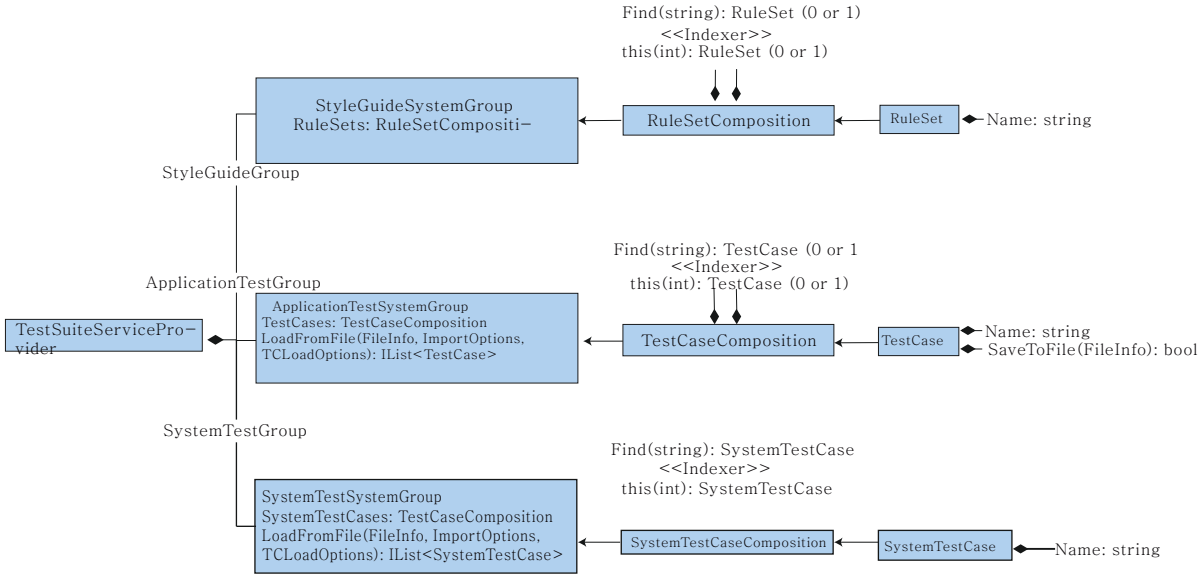
Functions

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

Test Suite Openness object model

Overview

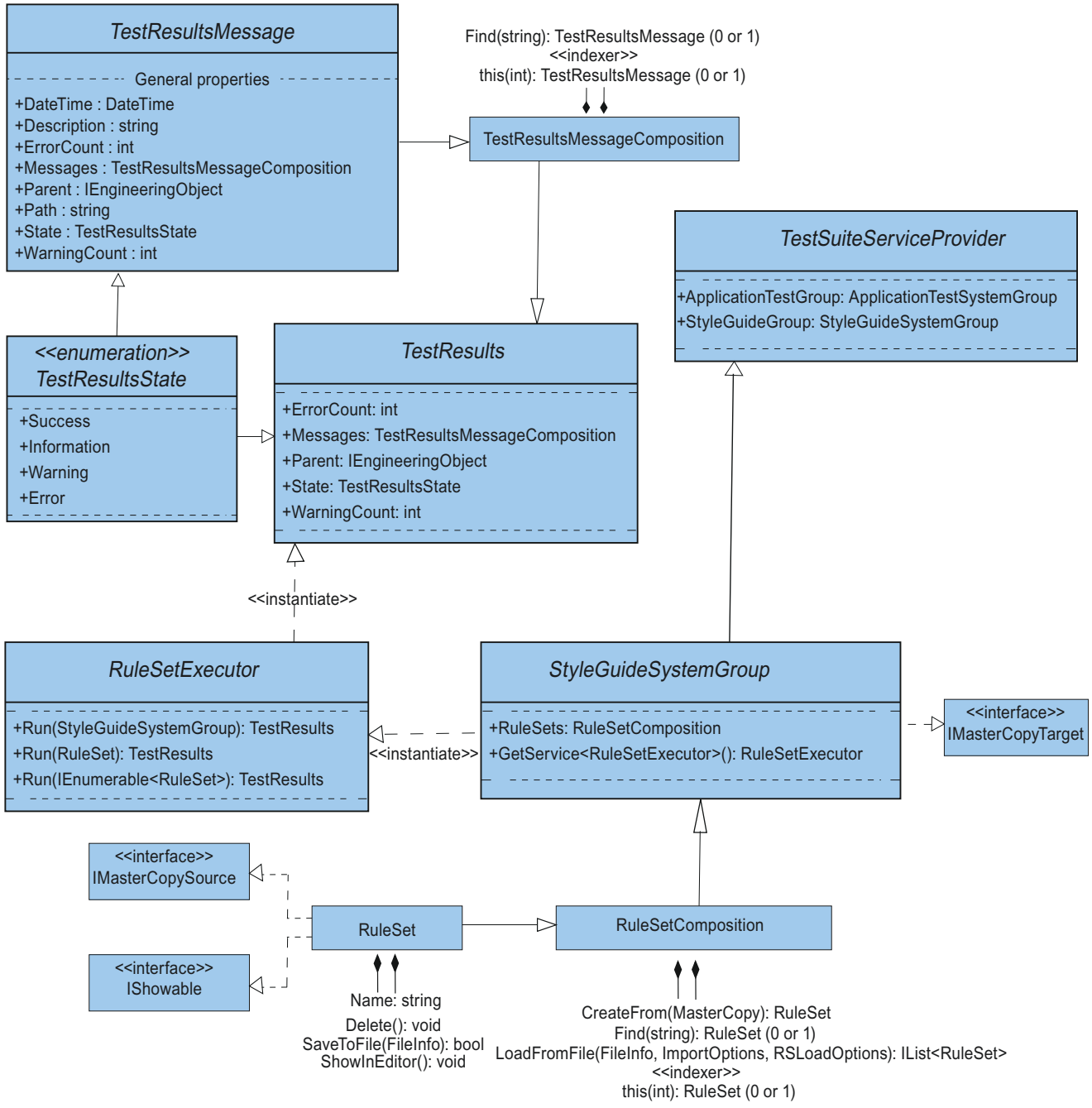
The following diagram describes the highest level of the Test Suite Openness object model:



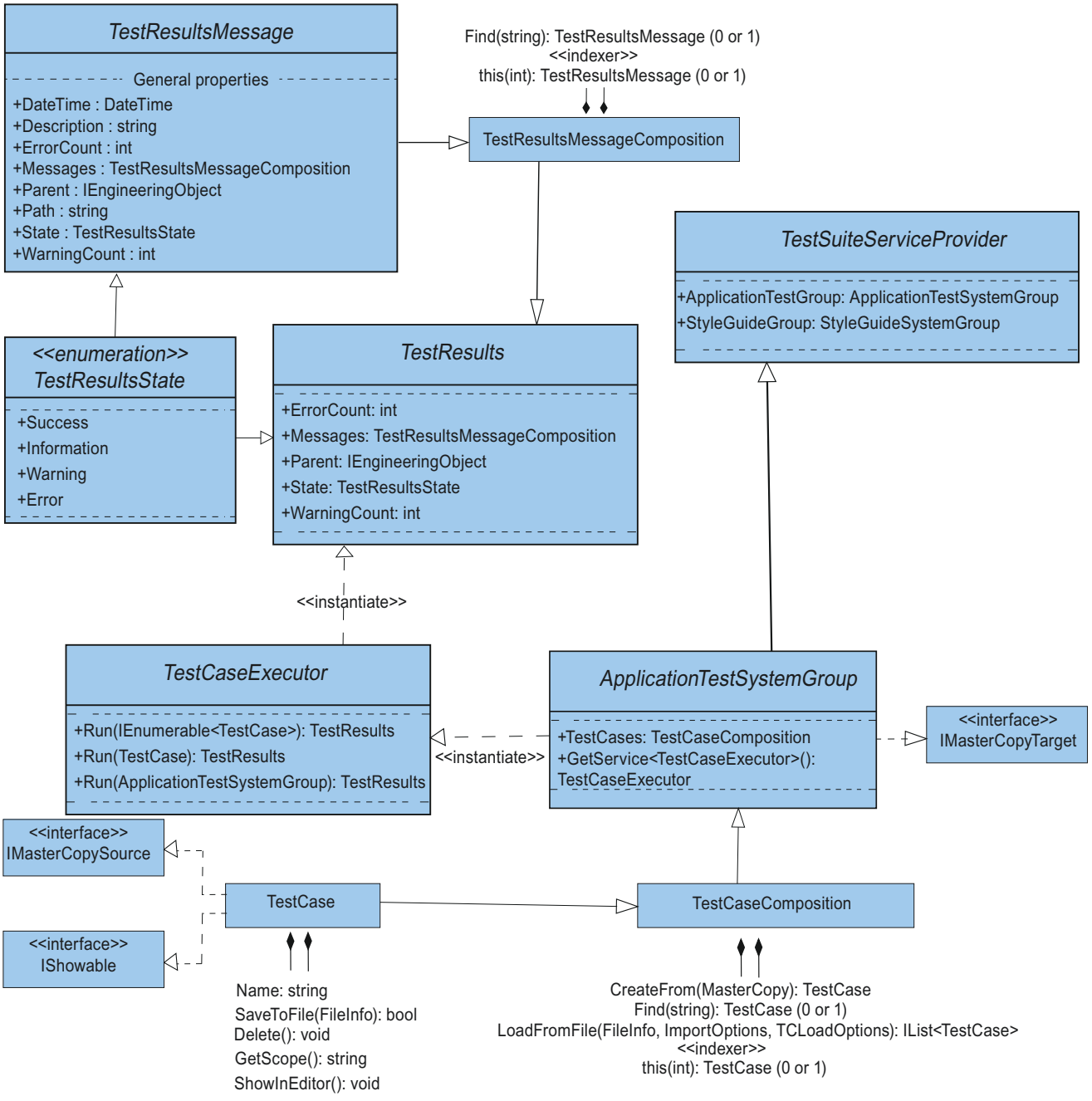
Class diagrams

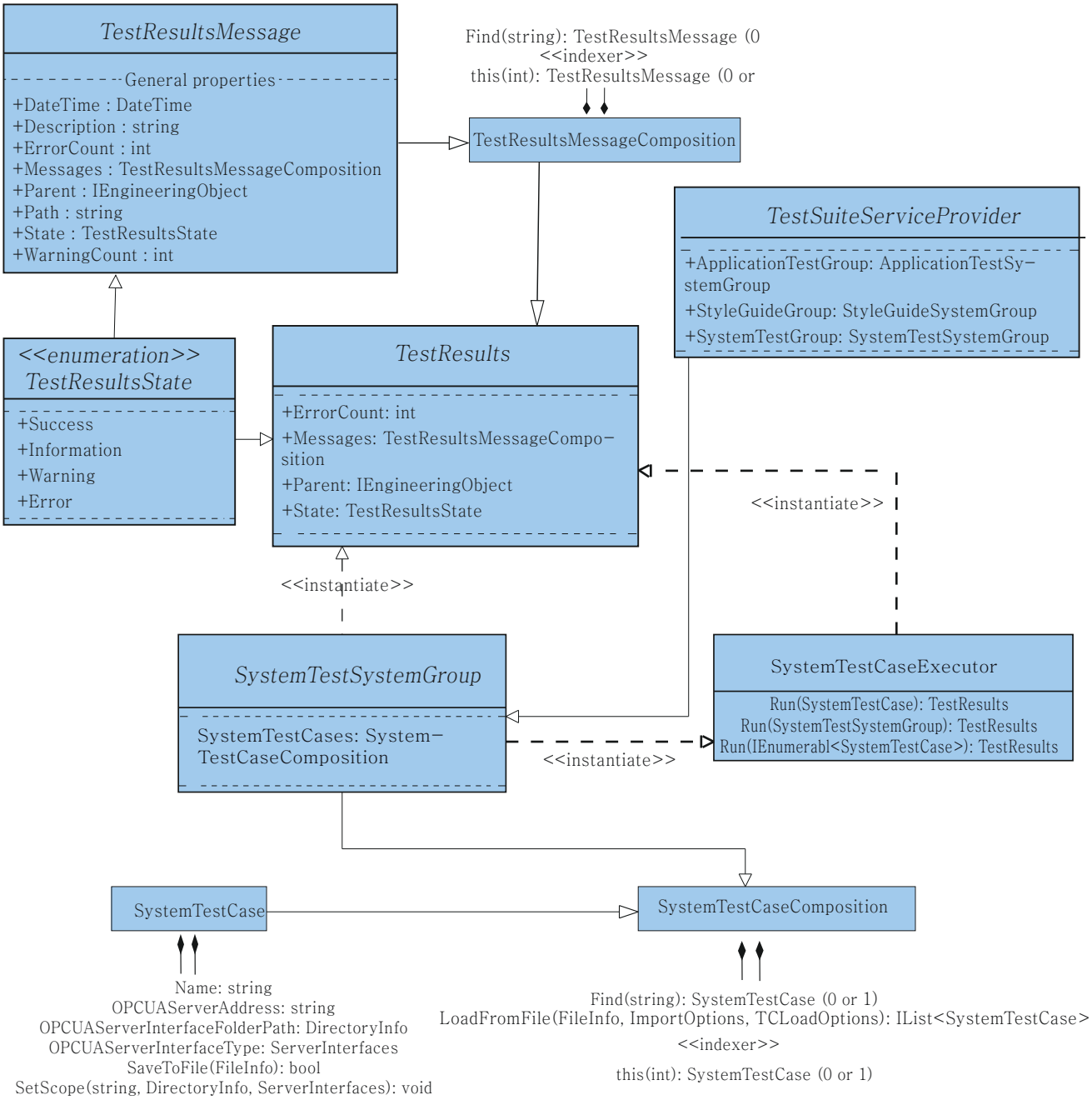
In the TIA Portal Openness object model all classes are defined as abstract which aren't directly instantiated.

The following diagram describes the objects which are located under Rule set execution and master copy.



The following diagram describes the object which are located under Test case execution and master copy.


TestResultsMessage
General properties
+DateTime : DateTime
+Description : string
+ErrorCount : int
+Messages : TestResultsMessageComposition
+Parent : IEngineeringObject
+Path : string
+State : TestResultsState
+WarningCount : int
Find(string): TestResultsMessage (0 or 1)
<<indexer>>
this(int): TestResultsMessage (0 or 1)
TestResultsMessageComposition
TestSuiteServiceProvider
+ApplicationTestGroup: ApplicationTestSystemGroup
+StyleGuideGroup: StyleGuideSystemGroup
<<enumeration>>
TestResultsState
+Success
+Information
+Warning
+Error
TestResults
+ErrorCount: int
+Messages: TestResultsMessageComposition
+Parent: IEngineeringObject
+State: TestResultsState
+WarningCount: int
<<instantiate>>
TestCaseExecutor
+Run(IEnumerable<TestCase>): TestResults
+Run(TestCase): TestResults
+Run(ApplicationTestSystemGroup): TestResults
<<instantiate>>
ApplicationTestSystemGroup
+TestCases: TestCaseComposition
+GetService<TestCaseExecutor>():
TestCaseExecutor
<<interface>>
IMasterCopyTarget
<<interface>>
IMasterCopySource
<<interface>>
IShowable
TestCase
Name: string
SaveToFile(FileInfo): bool
Delete(): void
GetScope(): string
ShowInEditor(): void
TestCaseComposition
CreateFrom(MasterCopy): TestCase
Find(string): TestCase (0 or 1)
LoadFromFile(FileInfo, ImportOptions, TCLoadOptions): IList<TestCase>
<<indexer>>
this(int): TestCase (0 or 1)


TestResultsMessage
General properties
+DateTime : DateTime
+Description : string
+ErrorCount : int
+Messages : TestResultsMessageComposition
+Parent : IEngineeringObject
+Path : string
+State : TestResultsState
+WarningCount : int
Find(string): TestResultsMessage (0
<<indexer>>
this(int): TestResultsMessage (0 or
TestResultsMessageComposition
TestSuiteServiceProvider
+ApplicationTestGroup: ApplicationTestSy-
stemGroup
+StyleGuideGroup: StyleGuideSystemGroup
+SystemTestGroup: SystemTestSystemGroup
<<enumeration>>
TestResultsState
+Success
+Information
+Warning
+Error
TestResults
+ErrorCount: int
+Messages: TestResultsMessageCompo-
sition
+Parent: IEngineeringObject
+State: TestResultsState
<<instantiate>>
<<instantiate>>
SystemTestSystemGroup
SystemTestCases: System-
TestCaseComposition
<<instantiate>>
SystemTestCaseExecutor
Run(SystemTestCase): TestResults
Run(SystemTestSystemGroup): TestResults
Run(IEnumerabl<SystemTestCase>): TestResults
SystemTestCase
SystemTestCaseComposition
Name: string
OPCUAServerAddress: string
OPCUAServerInterfaceFolderPath: DirectoryInfo
OPCUAServerInterfaceType: ServerInterfaces
SaveToFile(FileInfo): bool
SetScope(string, DirectoryInfo, ServerInterfaces): void
Find(string): SystemTestCase (0 or 1)
LoadFromFile(FileInfo, ImportOptions, TCLoadOptions): IList<SystemTestCase>
<<indexer>>
this(int): SystemTestCase (0 or 1)

4.2 Functions for Style guide

4.2.1 Accessing Style guide system folder

Program code

The following properties are used to access the system folder of a Style guide as system group:

```
Using Siemens.Engineering.TestSuite.StyleGuide;

StyleGuideSystemGroup scFolder = testSuite.StyleGuideGroup;
```

Reference

The following table describes the **properties** of the class for Style guide:

Property Name	Data Type	Access Mode
StyleGuideGroup	TestSuite.StyleGuide.StyleGuideSystemGroup	Read

4.2.2 Accessing Rule Set composition

Program code

Below property is used to list all available rule sets in the project:

```
RuleSetComposition listRuleSets = testSuite.StyleGuideGroup.RuleSets;
```

Reference

The following table describes the **properties** of the class for Style guide:

Property Name	Data Type	Access Mode
RuleSets	TestSuite.RuleSetComposition	Read

4.2.3 Accessing Rule Set

Description

Rule Sets in a TIA Portal project can be retrieved in two ways in openness application as below:

1. Indexed access on RuleSetComposition.
2. Using Find() method on RuleSetComposition.

Program code

Below properties are used to access the single rule set from the project and to retrieve a specific rule set from the project:

```
RuleSet rulesetDB = testSuite.StyleGuideGroup.RuleSets[1];
RuleSet ruleSet1 = testSuite.StyleGuideGroup.RuleSets.Find("rulesetDB");
```

Reference

The following table describes the **properties** of the class for Style guide:

Property Name	Data Type	Access Mode
RuleSet	RuleSetComposition	Read

Methods:

public RuleSet Find(string Name)

4.2.4 Accessing Rule Set name

Accessing Rule Set Name

The following property is used to access the rule set name:

```
string ruleSetName = testSuite.StyleGuideGroup.RuleSets[0].Name;
```

Renaming the selected Rule Set with a valid name

Below mentioned property is used to rename the selected rule set with a valid name.

```
public string Name { get; set; }
```

Note

You can perform the following possible tasks with the help of above "Name" property:

- Read the name of the existing rule set
- Change the name of the existing rule set

Reference

The following table describes the property of the class for Style guide:

Property Name	Data Type	Access Mode
Name	String	Read/Write

Program Code

You can modify the following program code to rename a rule set with a valid name:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
RuleSetExecutor styleGuideService =
testSuite.StyleGuideGroup.GetService<RuleSetExecutor>();

//Read the name of a rule set
//Retrieve single rule set
RuleSet myRuleSet1 = testSuite.StyleGuideGroup.RuleSets[0];
String myRuleSet_name = myRuleSet1.Name;

//Modify the name of a rule set
myRuleSet1.Name = "myRuleSet_modified";
```

Note

Application will display an exception if you provide wrong name (For example, starting with whitespace, or containing invalid character).

4.2.5 Show Rule Set

Open the selected Rule Set in "WithUI"

Below mentioned property is used to open the selected rule set in TIA Portal editor in the "WithUI" mode.

```
public void ShowInEditor();
```

Program Code

You can modify and use the following program code example to open the rule set in TIA Portal editor in the "WithUI" mode:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing rule set from same project
RuleSet myRuleSet = tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[1];

//Shows the selected item
myRuleSet.ShowInEditor();
```

4.2.6 Delete Rule Set

Delete the selected Rule Set in "WithUI/WithoutUI"

Below mentioned property is used to delete the selected rule set from the TIA Portal project in the "withUI/WithoutUI" mode.

```
public void Delete();
```

Program Code

You can modify and use the following program code example to delete the rule set from the TIA Portal project in the "withUI/WithoutUI" mode:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing rule set from same project
RuleSet myRuleSet = tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[1];

//Deletes the selected item
myRuleSet.Delete();
```

4.2.7 Modify the Scope of a Rule Set

Updating the Scope for existing Rule Set

Set the Scope of a Rule Set

You can use below API to update the scope of an existing rule set.

Reference

Scope of a rule set can be modified with the below methods:

- `SetScope()`
- `CopyScope()`

The following table describes the syntax of the class:

```
public bool SetScope(UpdateOptions updateOptions,
IList<IEngineeringObject> step7Objects);
```

The following table describes the properties of the class:

Name	Data type	Description
Parameter	`UpdateOptions.Override`	Overwrites the existing scope with the new list of objects available in same project
	`UpdateOptions.Add`	Appends the existing scope with the new list of objects available in same project
	`IList<IEngineeringObject>`	List of objects from the below table can be set as a scope for a rule set

Accessing Step7 objects to assign as a scope to the rule set

Step7 Objects	Representation in opennes application (IEngineeringObject)	Description
Program blocks	`PlcBlockSystemGroup`	All the available blocks under "Program blocks" folder will be set as scope except from "System blocks" folder
PLC tags	`PlcTagTableSystemGroup`	All the available tag tables under "PLC tags" folder will be set as scope
PLC data types	`PlcTypeSystemGroup`	All the available UDTs under "PLC data types" folder will be set as scope except under "System data types"

Software Units	`UnitGroup`
	`UnitGroup.Units[x]`
	`UnitGroup.Units[x].PlcBlockSystemGroup`
	`UnitGroup.Units[x].PlcTagTableSystemGroup`
	`UnitGroup.Units[x].PlcTypeSystemGroup`
		All the available Units under "Software Units" folder will be set as scope which includes Program blocks, PLC tags, and PLC data types. Specific selection is also possible inside Units folder

PLC/CPU	`DeviceItemClassifications.CPU / DeviceItems[0]`	Entire "PLC" with above objects will be set as scope
Project	`Project`	Entire "Project" with above objects will be set as scope
User groups	`PlcBlockSystemGroup.Groups[x]`	Specific user group under "Program blocks" folder can be set as scope
	`PlcTagTableSystemGroup.Groups[x]`	Specific user group under "PLC tags" folder can be set as scope
	`PlcTypeSystemGroup.Groups[x]`	Specific user group under "PLC data types" folder can be set as scope
	`Project.DeviceGroups[0]`	Group of PLCs can be selected as scope under project

Note

Step7 objects which are not supported by Style guide will be ignored during setting the scope and execution of a rule set.

Methods Used

Name of the method	Description	Return Value
`SetScope()`	To modify the scope of an existing rule set	`TRUE`: If the selected supported objects are set as scope
		`FALSE`: If the selected supported objects are not possible to set as scope

Program Code

You can modify and use the following program code example to retrieve the IEngineeringObjects available in the first device item:

```
using Siemens.Engineering;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

// Retrives the first device available in the project
var device = tiaProject.Devices.First();

// To retrieve “PLC_1” object from the above device:
var deviceItem = device.DeviceItems.First(e => e.Classification
==DeviceItemClassifications.CPU && e.Name == "PLC_1");

// To retrieve the software content of the “PLC_1”:
var plc = deviceItem.GetService<SoftwareContainer>().Software as PlcSoftware;

// To retrieve all the blocks available under “Program blocks” folder:
var plc1BlockGroup = plc.BlockGroup;

// To list blocks available under specific user group:
var plc1BlockGroupUserGroups = plc.BlockGroup.Groups[x];

// To provide all the tag tables available under “PLC tags” folder:
var plc1TagTableGroup = plc.TagTableGroup;

// To provide the PLC tag tables available under specific user group in “PLC Tags”
var plc1TagTableGroupUserGroups = plc.TagTableGroup.Groups[x];

// To provide all the UDTs available under "PLC data types" folder:
var plc1DataTypes = plc.TypeGroup;
var plc1DataTypes = plc.TypeGroup.Groups[x];

// To access all "Software units" available in “PLC_1”:
var plc1SoftwareUnit = plc.GetService<PlcUnitProvider>().UnitGroup;

// To provide all the supported objects available in "Entire Project":
ProjectBase Project1 = tiaProject;

// To provide objects from group of devices in a project:
var GroupOfPLC1 = tiaProject.DeviceGroups[x];

// To access rule set from same project
RuleSet ruleSet1 = tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[1];

// update the scope of a ruleSet1:
// The below line of code appends the existing scope with new list of objects:
ruleSet1.SetScope(UpdateOptions.Add, new List<IEngineeringObject> { plc1BlockGroup,
plc1DataTypes, plc1TagTableGroup, plc1SoftwareUnit });

// The below code overwrites the existing scope with new objects:
ruleSet1.SetScope(UpdateOptions.Override, new List<IEngineeringObject>{GroupOfPLC1 });
```

Assigning the Scope from one Rule Set to another Rule Set

You can use below API for copying the Scope from one rule set to another rule set in the same project.

Reference

The following table describes the syntax of the class:

```
public bool CopyScope(RuleSet targRuleSet);
```

The following table describes the properties of the class:

Name	Data type	Description
`targRuleSet`	`RuleSet`	Target rule set to update the scope

Methods used

Name of the method	Description	Return Value
`CopyScope()`	To assign the scope from one rule set to another rule set	`TRUE`: If the scope has been copied successfully
		`FALSE`: If the scope has been failed to copy

Program Code

You can modify and use the following program code example:

```
// To access source and target rule sets from same project:
RuleSet srcRuleSet = tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[1];
RuleSet targRuleSet =
tiaProject.GetService<TestSuiteService>().StyleGuideGroup.RuleSets[10];

RuleSet srcRuleSet1 =
tiaProject.GetService<TestSuiteService>().StyleGuideGroup.Find("myRS");
RuleSet targRuleSet1 =
tiaProject.GetService<TestSuiteService>().StyleGuideGroup.Find("PLC");

// To assign scope from source to target rule set:
bool retVal = srcRuleSet.CopyScope(targRuleSet);
bool retVal1 = srcRuleSet1.CopyScope(targRuleSet1);
```

4.2.8 Execution of Rule Sets

Rule set execution can be performed in three ways in openness application as mentioned below:

1. On specific rule set.
2. On list of rule sets.
3. All available rule sets in Style guide folder.

Reference

Signature

```
public RuleSetExecutor GetService<RuleSetExecutor>();
public TestResults Run(RuleSet ruleSet);
public TestResults Run(new IEnumerable<RuleSet>() {myRuleSet1,myStyleSet2,.. });
public TestResults Run(StyleGuideGroup allRuleSets);
```

The following attributes are supported by TestResults:

Attribute name	Data type	Access mode	Description
Messages	Siemens.Engineering.Test-Suite.TestResultsMessgeCom-position	Read	Composition of feed-back message
Parent	IEngineeringObject	Read	Returns the parent of the container
State	Siemens.Engineering.Test-Suite.TestResultsState	Read	Final state in a test re-sult feedback message with possibly values: Warning, Error and Suc-cess
WarningCount	Int	Read	Number of warnings in a test result feedback message
ErrorCount	Int	Read	Number of errors in a test result feedback message
Description	String	Read	Description or content of a test result feedback message
DateTime	DateTime	Read	Date and Time in a test result feedback mes-sage
Path	String	Read	Path to a test result feedback message

Methods used

Name of the method	Description
`Run()`	To run the selected rule sets

Note

Time Stamp Format

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

Program Code

You can modify and use the following program code example to:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

Project tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
RuleSetExecutor styleGuideService =
testSuite.StyleGuideGroup.GetService<RuleSetExecutor>();

//To execute specific rule set:
//Retrieve single rule set:
//Rule set at 0th index is available in the project:
RuleSet myRuleSet1 = testSuite.StyleGuideGroup.RuleSets[0];

//RuleSet_10 is available in the project:
RuleSet myStyleSet2 = testSuite.StyleGuideGroup.RuleSets.Find("RuleSet_10");

//Execute above retrieved rule sets:
TestResults ruleSetResults1 = styleGuideService.Run(myRuleSet1 );
TestResults ruleSetResults2 = styleGuideService.Run(myStyleSet2 );

//To execute list of rule sets:
//Retrieve list of rule sets to execute:
IEnumerable<RuleSet> ruleSetList = new List<RuleSet>() { myRuleSet1,myStyleSet2 };

//Execute above retrieved list of rule sets:
TestResults ruleSetsResults3 = styleGuideService.Run(ruleSetList );

//To execute all rule sets in StyleGuideGroup:
//Retrieve list of rule sets to execute:
StyleGuideSystemGroup styleGuideFolder = testSuite.StyleGuideGroup;

//Execute all available rule sets in Styleguide folder:
TestResults ruleSetsResults4 = styleGuideService.Run(styleGuideFolder);
```

User can access above test results and iterate through the TestResults messages using the following code:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;

...

//Retrieve test results of single rule set execution
WriteTestResults(ruleSetResults1);

//Retrieve test results of list of rule sets execution
WriteTestResults(ruleSetsResults3);

//Retrieve test results of rule sets execution available in
"StyleGuideGroup"
WriteTestResults(ruleSetsResults4);

...

private void WriteTestResults (TestResults result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);

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

RecursivelyWriteMessages(message.Messages, indent);
}
}
```

4.2.9 Import and Export of Rule Sets

Methods used

Following methods are used to import and export the rule sets:

Name of the method	Description
`SaveToFile()`	To export the rule set to an external source file
`LoadFromFile()`	To import the rule set(s) from external source file

Saving a rule set to an external file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `SaveToFile()` method on specific rule set to export it to a selected path.

Signature

```
public void SaveToFile(FileInfo path);
```

The following table shows the needed method parameters:

Name of the parameter	Description
`FileInfo`	Full Path for the rule set to be saved

Program code

Here, `styleChecks` and `styleChecks1` are rule sets available in "MyProject".

```
RuleSetComposition styleChecks = testSuite.StyleGuideGroup.RuleSets;
RuleSetComposition styleChecks1 = testSuite.StyleGuideGroup.RuleSets;
styleChecks.SaveToFile(exportFile);
styleChecks1.SaveToFile(new FileInfo(@"D:\Temp\TSRuleSets1.xml"));
```

Loading rule sets into project from external textual file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `LoadFromFile()` method on the `RuleSetComposition` to import rule set(s) from the selected file.

Signature

```
public IEnumerable<RuleSet> LoadFromFile (FileInfo path,
ImportOptions loadOptions, RSLoadOptions rsLoadOptions);
```

The following table shows the needed method parameters:

Name of the parameter	Description
FileInfo	Full Path for the rule set to be saved
ImportOptions.None	Throws exception if the rule set with the same name exists in the project/if the file contains multiple rule sets with the same name
ImportOptions.Override	Overwrites existing rule set in the project with the same name
RSLoadOptions.IgnorePropertyErrors	Imports the rule set(s) with invalid property values at Author/ Version/Comment
RSLoadOptions.None	Imports the rule set(s) only if there are no warnings/errors in the target file
RSLoadOptions.IgnoreMissingAttributes	Imports the rule set(s) with missing rule attributes by providing the default values to it
RSLoadOptions.SkipInvalidObjects	Imports the rule set(s) by skipping the rules with invalid attributes

Examples:

```
// Throw Exception if the rule set with the same name exists already
// Import the rule set with a invalid properties.

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions.None,
RSLoadOptions.IgnorePropertyErrors);
```

```
// Import rule set with the same name exists already
// Throw exception and abort the import of rule set when there are errors/warnings

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions.None, RSLoadOptions.None);
```

```
// Import rule set with the same name exists already

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions.Override, RSLoadOptions….);
```

```
// Overwrite rule set with the same name exists already in project

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions.Override, RSLoadOptions….);
```

```
// Import rule sets with the default values for the rules with missing attributes

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions…., 
RSLoadOptions.IgnoreMissingAttributes);
```

```
// Import rule sets and skip the import for the rules with invalid attributes

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions…., RSLoadOptions.SkipInvalidObjects);
```

```
// Import the rule set by replacing missing rule attributes with default values and skipping
rules with invalid attributes

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions…., RSLoadOptions.IgnoreMissingAttributes
| RSLoadOptions.SkipInvalidObjects);
```

```
// Imports the rule set(s) with errors in rule set property values, replaces the missing
rule attribute values with the default ones and skips the rules with invalid attributes

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSRuleSets1.xml");
styleChecks.LoadFromFile(LoadFile, ImportOptions…., RSLoadOptions.IgnoreMissingAttributes
| RSLoadOptions.SkipInvalidObjects | RSLoadOptions.IgnorePropertyErrors);
```

4.2.10 Working with Libraries

4.2.10.1 Copy Rule Set to Libraries

Rule sets from project can be copied to the libraries for the storage so that it can be reused and modified. And you don't explicitly need to export/import rule sets for external modification.

If the project library is opened in read-write mode, you can create a MasterCopy of an IMasterCopySource at target location.

Reference

The following table describes the syntax of the class:

```
public MasterCopy Create (IMasterCopySource sourceObject);
```

Methods Used

Name of the method	Description
`Create()`	To copy rule set from project to Project/Global libraries

Program Code

You can modify and use the following program code example to copy rule set from project to libraries:

```
using Siemens.Engineering;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;
using Siemens.Engineering.Library.Mastercopies;
...

// Retrieving rule set to copy to libraries:
RuleSet myRuleSet = testSuite.StyleGuideGroup.RuleSets.Find("CalcRuleSet");

// For copying a rule set to project library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Create(myRuleSet );

// For copying a rule set to project library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0]].MasterCopies.Create(myRuleSet );
tiaPortal.GlobalLibraries.Open(new FileInfo((@"D:\Documents\..), OpenMode.ReadWrite);

// For copying a rule set to global library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies.Create(myRuleSet );

// For copying a rule set to global library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies.Create(myRuleSet );
```

4.2.10.2 Copy Master copies to projects

The TIA Portal Openness API interface supports copying of master copies to project using the `CreateFrom` action. The action will create a new object based on the source master copy and place it in the composition where the action was called. The action will try to create the new rule set/rule sets with the same name as the source master copy. If such name is available in RuleSetComposition, the system will give the new rule set a new name. Then, it will return the new rule set copied.

Reference

The following table describes the syntax of the class:

```
public RuleSet CreateFrom (MasterCopy masterCopy);
```

Methods used

Name of the method	Description
`CreateFrom()`	To copy master copies from Project/Global libraries to project

Program Code

You can modify and use the following program code example to copying master copies from project to libraries:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.StyleGuide;
using Siemens.Engineering.Library.Mastercopies;

...

// For accessing master copy object in global library to copy to project:
var masterCopyObjectinGL = tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinGL =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies[1];

// To copy rule set from global library system folder to project:
RuleSet RuleSet1 =testSuite.StyleGuideGroup.RuleSets.CreateFrom(masterCopyObjectinGL);

// To copy rule set from global library user folder to project:
RuleSet RuleSet2
=testSuite.StyleGuideGroup.RuleSets.CreateFrom(masterCopyObjectinUserFolderinGL);

// For accessing master copy object in project library to copy to project:
var masterCopyObjectinPL =tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinPL
=tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0].MasterCopies[0];

// To copy rule set from project library system folder to project:
RuleSet RuleSet3 =testSuite.StyleGuideGroup.RuleSets.CreateFrom(masterCopyObjectinPL);

// To copy rule set from project library user folder to project:
RuleSet RuleSet4
=testSuite.StyleGuideGroup.RuleSets.CreateFrom(masterCopyObjectinUserFolderinPL);
```

4.2.10.3 Copy Master copies (of type RuleSet) within and between libraries

To copy master copies (of type TestCase) within and between libraries, refer to **TIA Portal Openness API** under chapter **Funtions on libraries** in the topic **Copying master copies** from the Information system.

4.3 Functions for Application Test

4.3.1 Accessing Application Test system folder

Program code

The following properties are used to access the system folder of a Test Suite as system group:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

TestSuiteService testSuite = project.GetService<TestSuiteService>( ) ;
ApplicationTestSystemGroup atfFolder = testSuite.ApplicationTestGroup;
```

Here 'project' is an instance of the TIA Portal project retrieved in the openness application.

Reference

The following table describes the **properties** of the class for application test:

Property Name	Data Type	Access Mode
ApplicationTestGroup	TestSuite.ApplicationTest.ApplicationTestSystemGroup	Read

4.3.2 Accessing Test Case composition

Program code

Below property is used to list all available test cases in the project:

```
TestCaseComposition listTestCases =
testSuite.ApplicationTestGroup.TestCases;
```

Reference

The following table describes the **properties** of the class for application test:

Property Name	Data Type	Access Mode
TestCases	TestSuite.TestCaseComposition	Read

4.3.3 Accessing Test Case

Description

Test cases in a TIA Portal project can be retrieved in two ways in openness application as below:

1. Indexed access on TestCaseComposition.
2. Using Find() method on TestCaseComposition.

Program code

Below properties are used to access the single test case from the project and to retrieve a specific test case from the project:

```
TestCase motorFCTestcase = testSuite.ApplicationTestGroup.TestCases[1];
TestCase testCase = testSuite.ApplicationTestGroup.TestCases.Find("TCMot");
```

Reference

The following table describes the **properties** of the class for application test:

Property Name	Data Type	Access Mode
TestCases	TestCaseComposition	Read

Methods:

public TestCase Find(string Name)

4.3.4 Accessing Test Case name

Accessing Test Case Name

The following property is used to access the test case name:

```
string testCaseName = testSuite.ApplicationTestGroup.TestCases[0].Name;
```

Renaming the selected Test Case with a valid name

Below mentioned property is used to rename the selected test case with a valid name.

```
public string Name { get; set; }
```

Note

You can perform the following possible tasks with the help of above "Name" property:

- Read the name of the existing test case
- Change the name of the existing test case

Reference

The following table describes the property of the class for application test:

Property Name	Data Type	Access Mode
Name	String	Read/Write

Program Code

You can modify the following program code to rename a test case with a valid name:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

ProjectBase tiaProject = … ;  TestSuiteService testSuite =
tiaProject.GetService<TestSuiteService>();
TestCaseExecutor applicationTestService =
testSuite.ApplicationTestGroup.GetService<TestCaseExecutor>();

//Read the name of a test case
//Retrieve single test case
TestCase myTestCase1 = testSuite.ApplicationTestGroup.TestCases[0];
String myTestCase_name = myTestCase1.Name;

//Modify the name of a test case
myTestCase1.Name = "myTestCase_modified";
```

Note

Application will display an exception if you provide wrong name (For example, starting with whitespace, or containing invalid character).

4.3.5 Accessing Test Case Scope

Accessing the Scope of existing Test Case

The following method is used to access the scope of an existing test case:

`GetScope()`

The following table describes the syntax of the class:

```
public plcSoftware GetScope();
```

The following table describes the property of the class:

Name	Data type	Description
Parameter	`string`	Name of the PLC, selected as test case scope

Program Code

You can modify and use the following program code example to access the existing test case scope:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing test case from same project
TestCase myTestCase =
tiaProject.GetService<TestSuiteService>().ApplicationTestGroup.TestCases[1];

//Access the scope of myTestCase
PlcSoftware twoHandMonitor = myTestCase.GetScope();
```

4.3.6 Show Test Case

Open the selected Test Case in "WithUI"

Below mentioned property is used to open the selected test case in TIA Portal editor in the "WithUI" mode.

```
public void ShowInEditor();
```

Program Code

You can modify and use the following program code example to open the test case in TIA Portal editor in the "WithUI" mode:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing test case from same project
TestCase myTestCase =
tiaProject.GetService<TestSuiteService>().ApplicationTestGroup.TestCases[1];

//Shows the selected item
myTestCase.ShowInEditor();
```

4.3.7 Delete Test Case

Delete the selected Test Case in "WithUI/WithoutUI"

Below mentioned property is used to delete the selected test case from the TIA Portal project in the "withUI/WithoutUI" mode.

```
public void Delete();
```

Program Code

You can modify and use the following program code example to delete the test case from the TIA Portal project in the "withUI/WithoutUI" mode:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Retrives "PLC_1" object from the above device
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");

//Accessing test case from same project
TestCase myTestCase =
tiaProject.GetService<TestSuiteService>().ApplicationTestGroup.TestCases[1];

//Deletes the selected item
myTestCase.Delete();
```

4.3.8 Modify the Scope of a Test Case

Updating the Scope for existing Test Case

Set the Scope of a Test Case

You can use below API to update the scope of an existing test case.

Reference

Scope of a test case can be modified with the method below:

- `SetScope()`

The following table describes the syntax of the class:

```
public void SetScope(plcSoftware PLC);
public void SetScope(plcSoftware PLC, string instanceName, ExecutionMode tcExecution);
```

The following table describes the properties of the class:

Parameter	Data type	Description
plcSoftware	`plcSoftware`	CPU types are valid objects which are compatible with PLCSIM Advanced
instanceName	`string`	Name of the PLC Advanced Instance
tcExecutionMode	`ExecutionMode`	To select test case execution mode with ExternallyManagedPLCSIMInstance / SystemManagedPLCSIMInstance

Note

It is not possible to set the scope for software controller using SetScope() API

Program Code

You can modify and use the following program code example:

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//To retrieve “PLC_1” object from the above device:
var deviceItem = device.DeviceItems.First(e => e.Classification
==DeviceItemClassifications.CPU && e.Name == "PLC_1");

//To retrieve the software content of the “PLC_1”:
var twoHandMonitor = deviceItem.GetService<SoftwareContainer>().Software
as PlcSoftware;

//To access test case from same project
TestCase myTestCase =
tiaProject.GetService<TestSuiteService>().ApplicationTestGroup.TestCases[1
];

//Updates the scope of myTestCase:
myTestCase.SetScope(twoHandMonitor);
```

```
using Siemens.Engineering;
using Siemens.Engineering.SW;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

//Retrives the first device available in the project
var device = tiaProject.Devices.First();

//Creates a new TIA portal instance
TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface);

//Assigns the path of the project to the project path variable
FileInfo projectPath = new FileInfo(project_path);

//Opens the TIA project AND Test Suite
Project tiaProject = tiaPortal.Projects.Open(projectPath);
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();

//To retrieve “PLC_1” object from the above device:
var deviceItem = device.DeviceItems.First(e => e.Classification ==
DeviceItemClassifications.CPU && e.Name == "PLC_1");
//To retrieve the software content of the “PLC_1”:
var plc = deviceItem.GetService<SoftwareContainer>().Software as
PlcSoftware;

//Specifies the instance name
string instanceName = "SetScope";

//To access test case from same project
TestCase myTestcase = testSuite.ApplicationTestGroup.TestCases[1];

//Updates the scope of ExternallyManagedPLCSIMInstance:
myTestcase.SetScope(plc, instanceName,
ExecutionMode.ExternallyManagedPLCSIMInstance);
```

4.3.9 Execution of Test Cases

Test case execution can be performed in three ways in openness application as mentioned below:

1. On specific test case.
2. On list of test cases.
3. All available test cases in Application test folder.

Reference

Signature

```
public TestCaseExecutor GetService<TestCaseExecutor>();
public TestResults Run(TestCase myTestCase);
public TestResults Run(new IEnumerable<TestCase>() {myTestCase1,myTestCase2,.. });
public TestResults Run(ApplicationTestGroup allTestCases);
```

The following attributes are supported by TestResults:

Attribute name	Data type	Access mode	Description
Messages	Siemens.Engineering.TestSuite.TestResultsMessgeComposition	Read	Composition of feedback message
Parent	IEngineeringObject	Read	Returns the parent of the container
State	Siemens.Engineering.TestSuite.TestResultsState	Read	Final state in a test result feedback message with possibly values: Warning, Error and Success
WarningCount	Int	Read	Number of warnings in a test result feedback message
ErrorCount	Int	Read	Number of errors in a test result feedback message
Description	String	Read	Description or content of a test result feedback message
DateTime	DateTime	Read	Date and Time in a test result feedback message
Path	String	Read	Path to a test result feedback message

Methods used

Name of the method	Description
`Run()`	To run the selected test case(s)

Note

Time Stamp Format

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

Note

You can run the test cases in both ExternallyManagedPLCSIMInstance or SystemManagedPLCSIMInstance configured mode of test case execution.

For information on ExternallyManagedPLCSIMInstance refer Test Case Execution using ExternallyManagedPLCSIMInstance (Page 58)

Program Code

You can modify and use the following program code example:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
TestCaseExecutor applicationTestService =
testSuite.ApplicationTestGroup.GetService<TestCaseExecutor>();

//To execute specific test case:
//Retrieve single test case:
//Test case at 0th index is available in the project:
TestCase myTestCase1 = testSuite.ApplicationTestGroup.TestCases[0];

//TestCase_10 is available in the project:
TestCase myTestCase2 =
testSuite.ApplicationTestGroup.TestCases.Find("TestCase_10");

//Execute above retrieved test cases:
TestResults TestCaseResults1 = applicationTestService.Run(myTestCase1 );
TestResults TestCaseResults2 = applicationTestService.Run(myTestCase2 );

//To execute list of test cases:
//Retrieve list of test cases to execute:
IEnumerable<TestCase> TestCaseList = new List<TestCase>()
{ myTestCase1,myTestCase2 };

//Execute above retrieved list of test cases:
TestResults TestCasesResults3 = applicationTestService.Run(TestCaseList );

//To execute all test cases in ApplicationTestGroup:
//Retrieve list of test cases to execute:
ApplicationTestSystemGroup applicationTestFolder =
testSuite.ApplicationTestGroup;

//Execute all available test cases in ApplicationTestFolder:
TestResults TestCasesResults4 =
applicationTestService.Run(applicationTestFolder);
```

User can access above test results and iterate through the TestResults messages using the following code:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

...

//Retrieve test results of single test case execution
WriteTestResults(TestCaseResults1);

//Retrieve test results of list of test cases execution
WriteTestResults(TestCasesResults3);

//Retrieve test results of test cases execution available in "ApplicationTestGroup"
WriteTestResults(TestCasesResults4);

...

private void WriteTestResults (TestResults result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);

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

RecursivelyWriteMessages(message.Messages, indent);
}
}
```

You can copy and modify the below sample code to run test cases in "ExternallyManagedPLCSIMInstance mode".

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;
using Siemens.Simatic.Simulation.Runtime;

string m_PlcInstanceName;
IInstance m_PlcSimInstance;



//To create PLCSIM Advanced instance and power on

m_PlcSimInstance = SimulationRuntimeManager.RegisterInstance(ECPUType.CPU1500_Unspecified,
m_PlcInstanceName);
m_PlcSimInstance.PowerOn(60000);

ProjectBase tiaProject = … ;

//Download the selected device item (Project PLC to PLCSIM instance created)

//Refer the chapter TIA Portal Openness API->Functions for accessing the data of a PLC
device->Functions for downloading data to PLC device ->Downloading to PLC devices

DeviceItem deviceItem = ...;
DownloadProvider downloadProvider = deviceItem.GetService<DownloadProvider>();
      if (downloadProvider != null)
                {
                      ...

                 }

//To retrieve the software content of the selected PLC to set the scope of test cases to run

var twoHandMonitor = deviceItem.GetService<SoftwareContainer>().Software as PlcSoftware;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
TestCaseExecutor applicationTestService =
testSuite.ApplicationTestGroup.GetService<TestCaseExecutor>();

//To run list of test cases:

//Test case at 0th index is available in the project:

TestCase myTestCase1 = testSuite.ApplicationTestGroup.TestCases[0];
myTestCase1.SetScope(twoHandMonitor,m_PlcInstanceName,ExecutionMode.ExternallyManagedPLCSI
MInstance);

//TestCase_10 is available in the project:

TestCase myTestCase2 = testSuite.ApplicationTestGroup.TestCases.Find("TestCase_10");
myTestCase2.SetScope(twoHandMonitor,m_PlcInstanceName,ExecutionMode.ExternallyManagedPLCSI
MInstance);

//Retrieve list of test cases to execute:

IEnumerable<TestCase> TestCaseList = new List<TestCase>() { myTestCase1,myTestCase2 };
```

```
//Execute above retrieved list of test cases, Connects to the exisiting PLCSIM Advanced
instance and runs the test cases selected

TestResults TestCasesResults = applicationTestService.Run(TestCaseList );
```

Working with PLCSIM Advanced V6.0

Working with TIA Portal in legacy mode

You can modify and use the following program code example:

```
//Initialize TIA Portal instance
var tiaPortalUi = new TiaPortal(TiaPortalMode.WithUserInterface);

//Open the project
var project = tiaPortalUi.Projects.Open(new FileInfo(projectPath));

//Enable legacy mode for the PLC

//For R/H type handling
var onlineProvider = device.GetService<RHOnlineProvider>();
var configuration = onlineProvider.Configuration;
if(!configuration.EnableLegacyCommunication) configuration.EnableLegacyCommunication =
true;

//For any other CPU type
var onlineProvider = deviceItem.GetService<OnlineProvider>();
var configuration = onlineProvider.Configuration;
if(!configuration.EnableLegacyCommunication) configuration.EnableLegacyCommunication =
true;

//Execute test case
var testSuiteService = project.GetService<TestSuiteService>();
var testCaseExecutor =
testSuiteService.ApplicationTestGroup.GetService<TestCaseExecutor>();
testCaseExecutor.Run(...);
```

Working with TIA Portal in secure mode

You can modify and use the following program code example:

```
//Initialize TIA Portal instance
var tiaPortalUi = new TiaPortal(TiaPortalMode.WithUserInterface);

//Open the project
var project = tiaPortalUi.Projects.Open(new FileInfo(projectPath));

//Enable secure mode for the PLC

//For R/H type handling
var onlineProvider = device.GetService<RHOnlineProvider>();
onlineProvider.Configuration.OnlineLegitimation += Configuration_OnlineLegitimation;

private void Configuration_OnlineLegitimation(OnlineConfiguration onlineConfiguration)
{
var tlsCommunication = onlineConfiguration as TlsVerificationConfiguration;
if(tlsCommunication == null) return;
tlsCommunication.CurrentSelection = TlsVerificationConfigurationSelection.Trusted;
}

//For any other CPU type
var onlineProvider = project.Devices[0].DeviceItems[1].GetService<OnlineProvider>();
onlineProvider.Configuration.OnlineLegitimation += Configuration_OnlineLegitimation;

private void Configuration_OnlineLegitimation(OnlineConfiguration onlineConfiguration)
{
var tlsCommunication = onlineConfiguration as TlsVerificationConfiguration;
if(tlsCommunication == null) return;
tlsCommunication.CurrentSelection = TlsVerificationConfigurationSelection.Trusted;
}
```

4.3.10 Import/Export of test cases using openness APIs

Methods used

Following methods are used to import and export the test cases:

Name of the method	Description
`SaveToFile()`	To export the test case
`LoadFromFile()`	To import the test case(s)

Saving a test case to an external textual file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `SaveToFile()` method on specific test case to export it to a selected path.

Signature

```
public void SaveToFile(FileInfo path);
```

The following table shows the needed method parameters:

Name of the parameter	Description
`FileInfo`	Full Path for the test case to be saved

Program code

Here, `motorFBTestCase` and `motorFBTestCase1` are test cases available in "MyProject".

```
motorFBTestCase.SaveToFile(exportFile);
motorFBTestCase1.SaveToFile(new FileInfo(@"D:\Temp\TSTestcases2.tat"));
```

Loading test cases into project from external textual file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `LoadFromFile()` method on the `TestCaseComposition` to import a test case(s) from the selected file.

Signature

```
public IEnumerable<TestCase> LoadFromFile (FileInfo path,
ImportOptions loadOptions, TCLoadOptions tcLoadOptions);
```

The following table shows the needed method parameters:

Name of the parameter	Description
`FileInfo`	Full Path for the test case to be saved
`ImportOptions.None`	Throws exception if the test case with the same name exists in the project/if the file contains a test case with the same name more than once
`ImportOptions.Override`	Overwrites existing test case in the project with the same name
`tcLoadOptions.IgnoreInvalidObject`	Import the test case if the scope/property in the file is not valid
`tcLoadOptions.None`	Throws exception if the test case imports with invalid scope/ invalid properties

Program code

Here, `motorFBTestCases` is an instance of a `TestCaseComposition`.

Examples:

```
// Load test cases from above obtained file path to project
// Throw Exception if the test case with the same name exists already
// Import the test case with a invalid scope/properties

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tat");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.None,
TCLoadOptions.IgnoreInvalidObject);
```

```
// Load test cases from above obtained file path to project
// Throw Exception if the test case with the same name exists already
// Throw Exception if the test case created with a invalid scope/invalid properties

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tat");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.None, TCLoadOptions.None);
```

```
// Load test cases from above obtained file path to project
// Import test case with the same name exists already
// Import the test case with a invalid scope/properties

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tat");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.Override,
TCLoadOptions.IgnoreInvalidObject);
```

```
// Load test cases from above obtained file path to project
// Overwrite test case with the same name exists already in project
// Throw Exception if the test case created with a invalid scope /invalid properties

FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tat");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.Override, TCLoadOptions.None);
```

4.3.11 Working with Libraries

4.3.11.1 Copy Test Case to Libraries

Test cases from project can be copied to the libraries for the storage so that it can be reused and modified. And you don't explicitly need to export/import test cases for external modification.

If the project library is opened in read-write mode, you can create a MasterCopy of an IMasterCopySource at target location.

Reference

The following table describes the syntax of the class:

```
public MasterCopy Create (IMasterCopySource sourceObject);
```

Methods Used

Name of the method	Description
`Create()`	To copy test case from project to Project/Global libraries

Program Code

You can modify and use the following program code example to copy test case from project to libraries:

```
using Siemens.Engineering;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;
...

// Retrieving test case to copy to libraries:
TestCase myTestCase = testSuite.ApplicationTestGroup.TestCases.Find("CalcTestCase");

// For copying a test case to project library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Create(myTestCase );

// For copying a test case to project library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.Folders[x].MasterCopies.Create(myTestCase );
tiaPortal.GlobalLibraries.Open(new FileInfo((@"D:\Documents\.."), OpenMode.ReadWrite);

// For copying a test case to global library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies.Create(myTestCase );

// For copying a test case to global library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[x].MasterCopies.Create(myTestCase );
```

4.3.11.2 Copy Master copies to projects

The TIA Portal Openness API interface supports copying of master copies to project using the `CreateFrom` action. The action will create a new object based on the source master copy and place it in the composition where the action was called. The action will try to create the new test case/test cases with the same name as the source master copy. If such name is available in TestCaseComposition, the system will give the new test case a new name. Then, it will return the new test case copied.

Reference

The following table describes the syntax of the class:

```
public TestCase CreateFrom (MasterCopy masterCopy);
```

Methods used

Name of the method	Description
`CreateFrom()`	To copy master copies from Project/Global libraries to project

Program Code

You can modify and use the following program code example to copying master copies from project to libraries:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.ApplicationTest;

...

// For accessing master copy object in global library to copy to project:
var masterCopyObjectinGL = tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinGL =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies[1];

// To copy test case from global library system folder to project:
TestCase testCase1
=testSuite.ApplicationTestGroup.TestCases.CreateFrom(masterCopyObjectinGL);

// To copy test case from global library user folder to project:
TestCase testCase2
=testSuite.ApplicationTestGroup.TestCases.CreateFrom(masterCopyObjectinUserFolderinGL);

// For accessing master copy object in project library to copy to project:
var masterCopyObjectinPL =tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinPL
=tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0].MasterCopies[0];

// To copy test case from project library system folder to project:
TestCase testCase3
=testSuite.ApplicationTestGroup.TestCases.CreateFrom(masterCopyObjectinPL);

// To copy test case from project library user folder to project:
TestCase testCase4
=testSuite.ApplicationTestGroup.TestCases.CreateFrom(masterCopyObjectinUserFolderinPL);
```

4.3.11.3 Copy Master copies (of type TestCase) within and between libraries

To copy master copies (of type TestCase) within and between libraries, Refer to **TIA Portal Openness API** under chapter **Funtions on libraries** in the topic **Copying master copies** from the Information system.

4.4 Functions for System Test

4.4.1 Accessing the System Test system folder

Program code

The following properties are used to access the system folder of a System test as system group:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
TestSuiteService testSuite = project.GetService<TestSuiteService>( );
SystemTestSystemGroup stFolder = testSuite.SystemTestGroup;
```

Here 'project' is an instance of the TIA Portal project retrieved in the openness application.

Reference

The following table describes the **properties** of the class for system test:

Property Name	Data Type	Access Mode
SystemTestGroup	TestSuite.SystemTest.SystemTestSystemGroup	Read

4.4.2 Accessing Test Case Composition

Program code

Below property is used to list all available system test cases in the project:

```
SystemTestCaseComposition listTestCases = testSuite.SystemTestGroup.SystemTestCases;
```

Reference

The following table describes the **properties** of the class for system test:

Property Name	Data Type	Access Mode
SystemTestCases	TestSuite.SystemTestCaseComposition	Read/Write

4.4.3 Accessing System Test Case

Description

Test cases in a TIA Portal project can be retrieved in two ways in openness application as below:

1. Indexed access on SystemTestCaseComposition.
2. Using Find() method on SystemTestCaseComposition.

Program code

Below properties are used to access the single test case from the project and to retrieve a specific test case from the project:

```
SystemTestCase motorFCTestcase = testSuite.SystemTestGroup.SystemTestCases[1];
SystemTestCase testCase = testSuite.SystemTestGroup.SystemTestCases.Find("TCMot");
```

Reference

The following table describes the properties of the class for system test:

Property Name	Data Type	Access Mode
SystemTestCases	TestSuite.SystemTestCaseComposition	Read/Write

Methods:

```
public SystemTestCase Find(string Name)
```

4.4.4 Accessing System Test Case name

Accessing System Test Case Name

The following property is used to access the test case name:

```
string testCaseName = testSuite.SystemTestGroup.SystemTestCases[0].Name;
```

Renaming the selected Test Case with a valid name

Below mentioned property is used to rename the selected test case with a valid name.

```
public string Name { get; set; }
```

Note

You can perform the following possible tasks with the help of above "Name" property:
- Read the name of the existing test case
- Change the name of the existing test case

Reference

The following table describes the property of the class for system test:

Property Name	Data Type	Access Mode
Name	String	Read/Write

Program Code

You can modify the following program code to rename a test case with a valid name:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
SystemTestCaseExecutor systemTestService =
testSuite.SystemTestGroup.GetService<SystemTestCaseExecutor>();
//Retrieve single test case
SystemTestCase myTestCase1 = testSuite.SystemTestGroup.SystemTestCases[0];
//Read the name of a test case
String myTestCase_name = myTestCase1.Name;
```

4.4.5 Accessing System Test Case Scope

Accessing the Scope of existing Test Case

The selected test case scope can be retrieved using the below mentioned test case properties

Syntax is as follows:

```
public string OPCUAServerAddress { get;}
public DirectoryInfo OPCUAServerInterfaceFolderPath{ get;}
public ServerInterfaces OPCUAServerInterfaceType { get;}
```

The following table describes the parameters:

Name	Data Type	Description
OPCUAServerAddress	String	OPC UA server address configured to the test case
OPCUAServerInterfaceFolderPath	DirectoryInfo	Directory information for the server interface files mapped
OPCUAServerInterfaceType	Siemens.Engineering.TestSuite.SystemTest.ServerInterfaces	Server interface type (UserDefined / standardSIMATIC / SiOMECompanionSpecification) configured.

Program Code

You can modify and use the following program code example to access the existing system test case scope:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
SystemTestCase tcForModule1 = testSuite.SystemTestGroup.SystemTestCases[0];
//Retrieve the OPC UA server address configured for a test
casestring tcForModule1OPCUAServeraddress = tcForModule1.OPCUAServerAddress;
//Retrieve the directory information of the OPC UA server interfaces mapped for a test
caseDirectoryInfo tcForModule1ServerInterfaces =
tcForModule1.OPCUAServerInterfaceFolderPath;
//Retrieve the full folder path of the OPC UA server interfaces mapped for a test
casestring  tcForModule1ServerInterfacesPath = tcForModule1ServerInterfaces.FullName;
//Validate the folder path of the OPC UA server interfaces mapped for a test
caseDirectoryInfo directoryInfo=new DirectoryInfo(@"D:\TestSuite\SystemTest\PACKMLFiles");
Assert.AreEqual(systemTestCase.OPCUAServerInterfaceFolderPath, directoryInfo);
// Retrieve the OPC UA server interface selected for a test case
ServerInterfcaes tcForModule1ServerInterfaceType = tcForModule1.OPCUAServerInterfaceType;
```

4.4.6 Modify the Scope of a System Test Case

Set the Scope of a System Test Case

You can use below API to modify the scope of an existing system test case.

Syntax of the methods is as follows:

```
public void SetScope ( string OPCUAServerAddress, ServerInterfaces
OPCUAServerInterfaceType, DirectoryInfo OPCUAServerInterfaceFolderPath );
```

Here the scope can be set using the OPCUAServerAddress, OPCUA ServerInterfaceType and OPCUA Server Interface Folder Path.

As the folder path is not mandatory run the test case you can use the following syntax modify the scope without parameter *"OPCUAServerInterfaceFolderPath"*

```
public void SetScope ( string OPCUAServerAddress, ServerInterfaces
OPCUAServerInterfaceType);
```

The following table describes the parameters:

Name	Data Type	Description
OPCUAServerAddress	String	OPC UA server address configured to the test case
OPCUAServerInterfaceFolderPath	DirectoryInfo	Directory information for the server interface files mapped
OPCUAServerInterfaceType	Siemens.Engineering.TestSuite.SystemTest.ServerInterfaces	Server interface type (UserDefined / standardSIMATIC /SiOMECompanionSpecification) configured.

Note

You can also assign the scope of one test case to another test case. Refer to the following program code for further information.

Note

Test case will be automatically refreshed/updated when you modify the scope of the test case. For futher information refer System test case scope update (Page 71)

Program Code

You can modify and use the following program code example:

```
using Siemens.Engineering;
……..
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;

// Accessing System test case
SystemTestCase tcModule1 =
tiaProject.GetService<TestSuiteService>().SystemTestGroup.SystemTestCases[1];

//  OPC UA server address
string targetPLCasOPCUAServer = “opc.tcp://opcua.demo-setscope.com:51210/UA/SampleServer”;

//Folder path to OPC UA server interfaces stored
DirectoryInfo tcModule1ServerInterafcesPath = new
DirectoryInfo(@"D:\Machines\ModuleTest\ServreInterafces");

//Setting the scope of a selected test case
tcModule1.SetScope(targetPLCasOPCUAServer ,ServerInterfaces.StandardSIMATIC ,
tcModule1ServerInterafcesPath);

**********************************************************************
Assigning the scope of one test case to another test case
**********************************************************************

SystemTestCase tcForModule3 = testSuite.SystemTestGroup.SystemTestCases[3];
SystemTestCase tcForModule4 = testSuite.SystemTestGroup.SystemTestCases[4];

//Retrieve the OPC UA server address configured for a “tcForModule3”

string tcForModule3OPCUAServeraddress = tcForModule3.OPCUAServerAddress;

//Retrieve the directory information of the OPC UA server interfaces mapped for a
“tcForModule3”

DirectoryInfo tcForModule3ServerInterfaces = tcForModule3.OPCUAServerInterfaceFolderPath;

// Retrieve the OPC UA server interface selected for a “tcForModule3”

ServerInterfcaes tcForModule3ServerInterfaceType = tcForModule3.OPCUAServerInterfaceType;

//Setting the scope of a “tcForModule4”
tcForModule4.SetScope(tcForModule3OPCUAServeraddress, tcForModule3ServerInterfaceType,
tcForModule3ServerInterfaces);
```

Error message during scope modification

The following user scenarios and corresponding error messages are tabulated below.

User Scenarios	Error messages
“OPCUAServerAddress” is not mentioned in right format i.e.,opc.tcp://server:port/path	Specified 'OPCUAServerAddress' is invalid. Please use the format 'opc.tcp://server:port/path'
When null value is used for “OPCUAServerAddress”	'OPCUAServerAddress' cannot be null or empty. Please use the format 'opc.tcp://server:port/path'
Trying to update the scope of a test case in secondary project.	Error message should be from openness framework
No matching files exists in the specified “OPCUAServerInterfaceFolderPath”	Specified 'OPCUAServerInterfaceFolderPath ' does not contain valid server interface files
Folder path mentioned at “OPCUAServerInterfaceFolderPath” is not available	Cannot find directory '<Drive_letter>:\\'
Folder path mentioned at “OPCUAServerInterfaceFolderPath” contains illegal characters	Illegal characters in path
Folder path mentioned at “OPCUAServerInterfaceFolderPath” is unauthorized path	Cannot read file '<Folderpath>'
Folder path mentioned at “OPCUAServerInterfaceFolderPath” is Specific/relative path	The argument ‘<path>’ cannot be a specific path
Folder path mentioned at “OPCUAServerInterfaceFolderPath” contains additional whitespace characters (i.e. leading and trailing whitespace)	The argument '<path>' contains additional white-space characters which is not allowed.
Folder path mentioned at “OPCUAServerInterfaceFolderPath” from a disconnectable/removable path	Cannot find directory 'G:\\' for path 'G:\\RemovableUSB_Disk'
Folder path mentioned at “OPCUAServerInterfaceFolderPath” contains more than 248 characters	The specified path, file name, or both are too long.

4.4.7 Execution of System Test Cases

Execution of System Test Cases

Test case execution can be performed in three ways in openness application as mentioned below.

1. On specific test case.
2. On list of test cases.
3. All available test cases in System test folder.

Reference

Signature

```
public SystemTestCaseExecutor GetService<SystemTestCaseExecutor>();
public TestResults Run(SystemTestCase myTestCase);
public TestResults Run(new IEnumerable<SystemTestCase>() {myTestCase1,myTestCase2,.. });
public TestResults Run(SystemTestGroup allTestCases);
```

The following attributes are supported by TestResults:

Attribute name	Data type	Access mode	Description
Messages	Siemens.Engineering.Test-Suite.TestResults-MessgeComposition	Read	Composition of feedback message
Parent	IEngineeringObject	Read	Returns the parent ofthe container
State	Siemens.Engineering.Test-Suite.TestResults-State	Read	Final state in a test result feedback message with possibly values:Warning, Error and Success
Warning-Count	Int	Read	Number of warnings in a test result feedback message
ErrorCount	Int	Read	Number of errors in atest result feedback message
Description	String	Read	Description or contentof a test result feedback message
DateTime	DateTime	Read	Date and Time in a test result feedback message
Path	String	Read	Path to a test result feedback message

Methods Used:

Name of the method	Description
Run()	To run the selected test case(s)

Note

Time Stamp Format

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

Program Code

You can modify and use the following program code example:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
ProjectBase tiaProject = … ;
TestSuiteService testSuite = tiaProject.GetService<TestSuiteService>();
SystemTestCaseExecutor systemTestService =
testSuite.SystemTestGroup.GetService<SystemTestCaseExecutor>();
//To execute specific test case:
//Retrieve single test case:
//Test case at 0th index is available in the project:
SystemTestCase myTestCase1 = testSuite.SystemTestGroup.SystemTestCases[0];
//TestCase_10 is available in the project:
SystemTestCase myTestCase2 = testSuite.SystemTestGroup.SystemTestCases.Find("TestCase_10");
//Execute above retrieved test cases:TestResults TestCaseResults1 =
systemTestService.Run(myTestCase1);
TestResults TestCaseResults2 = systemTestService.Run(myTestCase2);
//To execute list of test cases://Retrieve list of test cases to execute:
IEnumerable<SystemTestCase> TestCaseList = new List<SystemTestCase>()
{ myTestCase1,myTestCase2 };
//Execute above retrieved list of test cases:
TestResults TestCasesResults3 = systemTestService.Run(TestCaseList);
//To execute all test cases in SystemTestGroup:
//Retrieve list of test cases to execute:
SystemTestSystemGroup systemTestFolder = testSuite.SystemTestGroup;
//Execute all available test cases in SystemTestFolder:
TestResults TestCasesResults4 = systemTestService.Run(systemTestFolder);
```

User can access above test results and iterate through the TestResults messages using the following code:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
...
//Retrieve test results of single test case execution
WriteTestResults(TestCaseResults1);
//Retrieve test results of list of test cases execution
WriteTestResults(TestCasesResults3);
//Retrieve test results of test cases execution available in "SystemTestGroup"
WriteTestResults(TestCasesResults4);
...
private void WriteTestResults (TestResults result)
{
Console.WriteLine("State:" + result.State);
Console.WriteLine("Warning Count:" + result.WarningCount);
Console.WriteLine("Error Count:" + result.ErrorCount);
RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(TestResultsMessageComposition messages,string
indent= "")
{
indent += "\t";
foreach (TestResultsMessage message in messages)
{
Console.WriteLine(indent + "Path: " + message.Path);
Console.WriteLine(indent + "DateTime: " + message.DateTime);
Console.WriteLine(indent + "State: " + message.State);
Console.WriteLine(indent + "Description: " + message.Description);
Console.WriteLine(indent + "Warning Count: " + message.WarningCount);
Console.WriteLine(indent + "Error Count: " + message.ErrorCount);
RecursivelyWriteMessages(message.Messages, indent);}}
```

4.4.8 Import/Export of system test cases using openness APIs

Methods used

The following methods are used to import and export the test cases:

Name of the method	Description
`SaveToFile()`	To export the system test case
`LoadFromFile()`	To import the system test case(s)

Saving a test case to an external textual file

Requirement

The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `SaveToFile()` method on specific test case to export it to a selected path.

Signature

```
public void SaveToFile(FileInfo path);
```

The following table shows the needed method parameters:

Name of the parameter	Description
FileInfo	Full Path for the system test case to be saved

Program Code

Here, `motorFBTestCase` and `motorFBTestCase1` are test cases available in "MyProject".

```
motorFBTestCase.SaveToFile(exportFile);
motorFBTestCase1.SaveToFile(new FileInfo(@"D:\Temp\TSTestcases2.tst"));
```

Loading test cases into project from external textual file

Requirement

The TIA Portal Openness application is connected to the TIA Portal.

Application

Use the `LoadFromFile()` method on the `SystemTestCaseComposition` to import a test case(s) from the selected file.

Signature

```
public IList<SystemTestCase> LoadFromFile (FileInfo
path,ImportOptions loadOptions, TCLoadOptions tcLoadOptions);
```

The following table shows the needed method parameters:

Name of the parameter	Description
`FileInfo`	Full Path for the test case to be saved
`ImportOptions.None`	Throws exception if the test case with the same name exists in the project
`ImportOptions.Override`	Overwrites existing test case in the project with the same name
`tcLoadOptions.IgnoreInvalidObject`	Imports the test case though the test case scope(OPC UA server address and interface selected /properties in the file is invalid/missing in the file)
`tcLoadOptions.None`	Throws exception if test case imports with invalid/missing OPC UA server address and also if file contains Interface_Selected/ test case properties are invalid.

Program code

Here, motorFBTestCases is an instance of a SystemTestCaseComposition.

Examples:

```
// Load test cases from above obtained file path to project
// Throw Exception if the test case with the same name exists already
// Import the test case with a invalid scope/properties
FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tst");
motorFBTestcases.LoadFromFile(LoadFile,
ImportOptions.None,TCLoadOptions.IgnoreInvalidObject);
// Load test cases from above obtained file path to project
// Throw Exception if the test case with the same name exists already
// Throw Exception if the test case created with a invalid/not available scope/invalid
properties
FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tst");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.None, TCLoadOptions.None);
// Load test cases from above obtained file path to project
// Import test case with the same name exists already
// Import the test case with a invalid scope/properties
FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tst");
motorFBTestcases.LoadFromFile(LoadFile,
ImportOptions.Override,TCLoadOptions.IgnoreInvalidObject);
// Load test cases from above obtained file path to project
// Overwrite test case with the same name exists already in project
// Throw Exception if the test case created with a invalid/not available scope/invalid
properties
FileInfo LoadFile = new FileInfo(@"D:\Temp\TSTestcases1.tst");
motorFBTestcases.LoadFromFile(LoadFile, ImportOptions.Override, TCLoadOptions.None);
```

4.4.9 Working with Libraries

4.4.9.1 Copy System Test Case to Libraries

System test cases from project can be copied to the libraries for storage and can be reused and modified. You do not have to explicitly export/import system test cases for external modification.

If the project library is opened in read-write mode, you can create a MasterCopy of an IMasterCopySource at target location.

Reference

The following table describes the syntax of the class:

```
public MasterCopy Create (IMasterCopySource sourceObject);
```

Methods Used

Name of the method	Description
`Create()`	To copy system test case from project to Project/ Global libraries

Program Code

You can modify and use the following program code example to copy test case from project to libraries:

```
using Siemens.Engineering;
...
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;
...

// Retrieving a system test case to copy to libraries:
SystemTestCase mySystemTestCase =
testSuite.SystemTestGroup.SystemTestCases.Find("CalcTestCase");

// For copying a system test case to project library master copy system
folder:
MasterCopy masterCopyInSystemFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Create(mySystemTes
tCase );

// For copying a system test case to project library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0].MasterCopies.Create(
mySystemTestCase );
tiaPortal.GlobalLibraries.Open(new FileInfo((@"D:\Documents\.."),
OpenMode.ReadWrite);

// For copying a test case to global library master copy system folder:
MasterCopy masterCopyInSystemFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies.Create(mySystem
TestCase );

// For copying a test case to global library master copy user folder:
MasterCopy masterCopyinUserFolder =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies.Crea
te(mySystemTestCase );
```

4.4.9.2 Copy Master copies to projects

The TIA Portal Openness API interface supports copying of master copies to project using the `CreateFrom` action. The action will create a new object based on the source master copy and place it in the composition where the action was called. The action will try to create the new system test case/s with the same name as the source master copy. If such name is available in SystemTestCaseComposition, the system will give the new test case a new name. Then, it will return the new system test case copied.

Reference

The following table describes the syntax of the class:

```
public SystemTestCase CreateFrom (MasterCopy masterCopy);
```

Methods used

Name of the method	Description
`CreateFrom()`	To copy master copies from Project/Global libraries to project

Program Code

You can modify and use the following program code example to copying master copies from project to libraries:

```
using Siemens.Engineering;
using Siemens.Engineering.TestSuite;
using Siemens.Engineering.TestSuite.SystemTest;

...

// For accessing master copy object in global library to copy to project:
var masterCopyObjectinGL =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinGL =
tiaPortal.GlobalLibraries[0].MasterCopyFolder.Folders[0].MasterCopies[1];

// To copy system test case from global library system folder to project:
SystemTestCase SystemtestCase1
=testSuite.SystemTestGroup.SystemTestCases.CreateFrom(masterCopyObjectinGL
);

// To copy test case from global library user folder to project:
SystemTestCase SystemtestCase2
=testSuite.SystemTestGroup.SystemTestCases.CreateFrom(masterCopyObjectinUs
erFolderinGL);

// For accessing master copy object in project library to copy to project:
var masterCopyObjectinPL
=tiaProject.ProjectLibrary.MasterCopyFolder.MasterCopies[0];
var masterCopyObjectinUserFolderinPL
=tiaProject.ProjectLibrary.MasterCopyFolder.Folders[0].MasterCopies[0];

// To copy system test case from project library system folder to project:
SystemTestCase SystemtestCase3
=testSuite.SystemTestGroup.SystemTestCases.CreateFrom(masterCopyObjectinPL
);

// To copy systemtest case from project library user folder to project:
SystemTestCase SystemtestCase4
=testSuite.SystemTestGroup.SystemTestCases.CreateFrom(masterCopyObjectinUs
erFolderinPL);
```

4.4.9.3 Copy Master copies (of type SystemTestCase) within and between libraries

To copy master copies (of type SystemTestCase) within and between libraries, Refer to **TIA Portal Openness API** under chapter **Funtions on libraries** in the topic **Copying master copies** from the Information system.

4.5 Exceptions

Handling exceptions

Exceptions when accessing the Test Suite via TIA Portal Openness API

During the execution of a Test Suite application via the TIA Portal Openness API, all errors which occur are reported as exceptions. These exceptions contain information that will help you to correct the errors which have occurred.

A distinction is made between two types of exceptions:

- Recoverable (Siemens.Engineering.EngineeringException)
  You can continue to access the TIA Portal without interruption with this exception. Alternatively, you can cancel the connection to the TIA Portal.
  The EngineeringExceptions include the following types:
  - Exceptions when executing test case (SupportSimulationNotEnabledException), when the support simulation property is not enabled in the project.
  - OPCUA Server invalid address exception (OPCUAServerAddressNotValidException) - This exception occurs when the executed test cases have and invalid OPCUA server address.
  - License not found exception (LicenseNotFoundException) - This exception occurs when a valid Test Suite license in not available
- NonRecoverable (Siemens.Engineering.NonRecoverableException)
  This exception closes the TIA Portal and the connection to the TIA Portal is disconnected. You need to restart the TIA Portal using the TIA Portal Openness application.