

## TIA Portal Test Suite

Function Manual

Valid for TIA Portal Test Suite

## 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.

# Table of contents

<b>1</b>	<b>Security information .....</b>	<b>4</b>
<b>2</b>	<b>Styleguide.....</b>	<b>5</b>
2.1	Introduction .....	5
2.2	Creating and Managing rule sets .....	6
2.3	Rule Types .....	10
2.3.1	Casing .....	10
2.3.2	Name Contains .....	14
2.3.3	Metadata .....	15
2.3.4	Name Length .....	17
2.3.5	Prefix Suffix .....	18
2.4	Object Selector .....	20
2.5	Display of test results .....	21
2.6	Additional Features .....	23
<b>3</b>	<b>Application Test .....</b>	<b>24</b>
3.1	Introduction .....	24
3.2	Creating and Managing Test Cases .....	26
3.3	Aliasing a Program Variable .....	29
3.4	Basic Instructions .....	31
3.5	Supported Data Types .....	35
3.5.1	Binary Numbers .....	35
3.5.2	Integers .....	36
3.5.3	Floating Point Numbers .....	38
3.5.4	Timers .....	39
3.5.5	Strings .....	39
3.6	Test Case Execution .....	40
3.7	Test Case Execution Failure .....	42
3.8	Additional Features .....	43

## 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/industrialsecurity>.

# Styleguide

## 2.1 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 DBs/Instance DBs
- 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

## 2.2 Creating and Managing rule sets

### Introduction

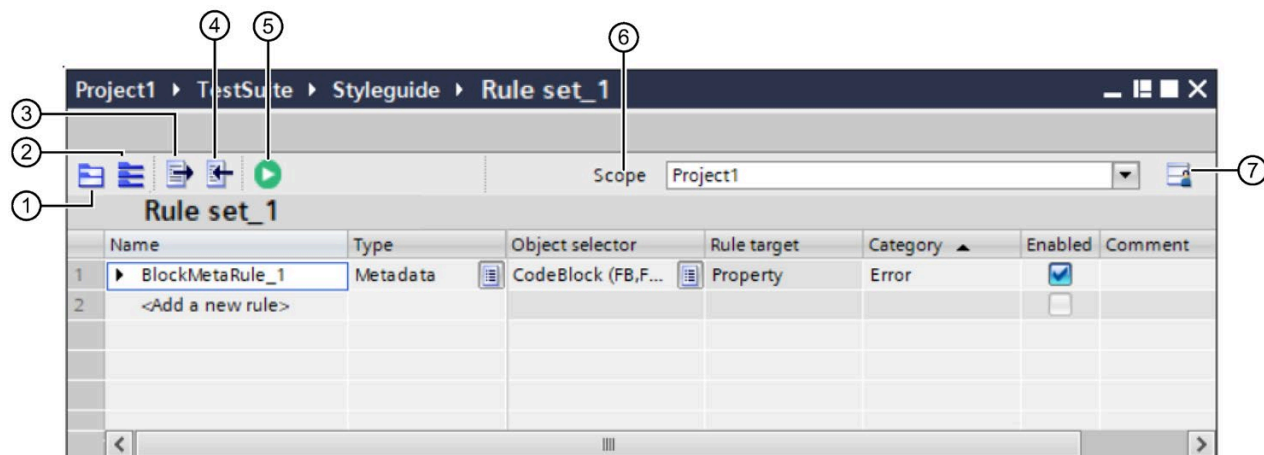
This chapter guides you to create and manage rule sets. It also assists you in performing general rule set operations.

### Rule set editor

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

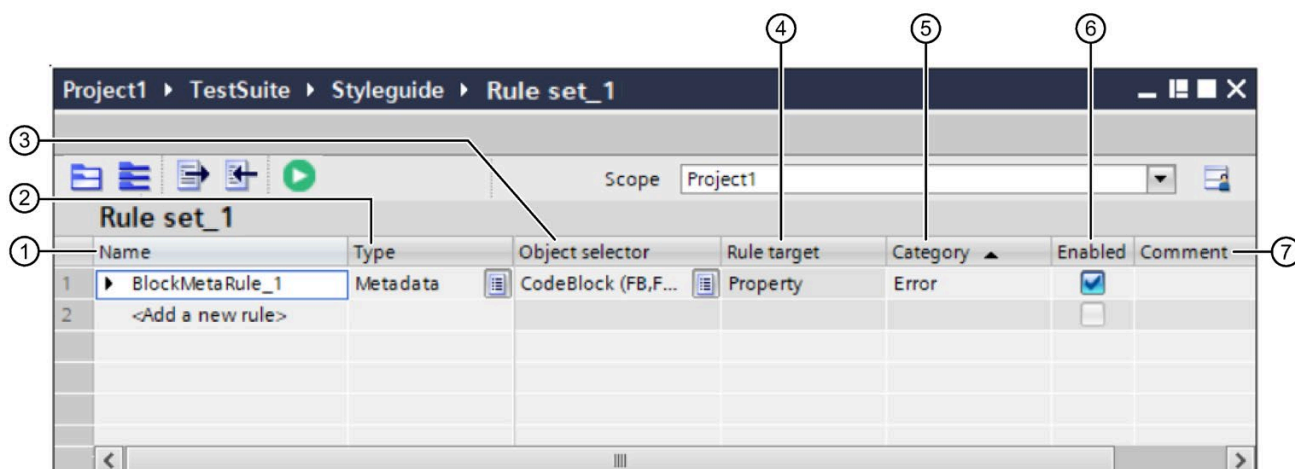
#### Structure of a rule set

##### 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 styleguide 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 V16 Update 1 installed.
- Optional Package TIA Portal Test Suite Advanced V16 is installed.

**Note**

As an example, we will use "Rule set\_1" throughout the styleguide chapter. The user can use any Rule set name when creating a styleguide.

### 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 PNV.

---

#### Note

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

### Deleting a rule set

You can delete a rule set from the PNV and deleting will remove the rule set from the project PNV under the "Styleguide: subfolder.

#### Procedure 1

1. Under the "Project tree" area, select "Project 1 > Test Suite > Styleguide > 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 PNV. 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 > TestSuite > Styleguide > Rule set\_1".
2. Right click "Rule set\_1 > Rename".
3. Rename the rule set according to your choice.

---

### Note

- A rule set name can accept a maximum of 125 unicode characters
  - When multiple rule sets are selected, you can rename the last selected rule set.
  - 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 rule set editor

You can open a rule set editor and edit it separately 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 PNV

You can copy rule set(s) from the PNV and copying them will add the rule set(s) to the respective project PNV under the "Style guide" subfolder with the name similar to PNV 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 rule sets from the PNV.
  - Rule sets are sorted automatically after you perform the paste operation based on the precedence of the unicode char
- 

## 2.3 Rule Types

### 2.3.1 Casing

#### Introduction

This feature allows you to perform a capitalization check for a particular type of PLC objects selected. This styleguide check will support 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 V16 UPD1 is open.
- Testsuite 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
  - Camel casing: First character is always in lower case. 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. 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 > TestSuite > Styleguide".
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" "\$&%CASING" "Casing1" "CaSiNd1234" "casing1234CASING" "123" "U" "UPPER_CASING" "upper"	Should Identify the objects "CASING" "CASING123" "\$&%CASING" "123" "U" "UPPER_CASING"
Upper Casing	Is Not Set	"CASING" "CASING123" "\$&%CASING" "Casing1" "CaSiNd1234" "casing1234CASING" "123" "U" "UPPER_CASING" "upper"	Should Identify the objects "Casing1" "CaSiNd1234" "casing1234CASING" "upper"
Pascal Casing	Is Set	"Casing" "CASING123" "\$&%Casing" "@Casing" "Casing1Casing2" "Casing1casing2" "CAasing1Casing2" "P" "P1" "PascalC" "PascalC1" "PascalC1Test" "123" "PA" "Pascal_Casing"	Should Identify the objects "Casing" "\$&%Casing" "@Casing" "Casing1Casing2" "P" "P1" "PascalC" "PascalC1" "PascalC1Test" "123" "Pascal_Casing"

Casing Type	Violation Condition	Object Name	Test Result
Pascal Casing	Is Not Set	" casing " " CASING123 " " \$&% casing " " @ casing " " casing1 casing2 " " casing1 casing2 " " Casing1 Casing2 " " P " " P1 " " PascalC " " PascalC1 " " PascalC1Test " " 123 " " PA " " Pascal_ Casing "	Should identify the objects " CASING123 " " casing1 casing2 " " Casing1 Casing2 "
Camel Casing	Is Set	" caSing " " caSing123 " " \$&% cAsing " " @ casing " " casing " " casinG1 casing2 " " casing1 Casing2 Casing3 " " cASING " " Casing " " camel1c " " c " " 123 " " camel_ Casing " " camelC " " camelC1 "	Should identify the objects " caSing " " caSing123 " " \$&cAsing " " @ casing " " casing " " casinG1 casing2 " " camel1c " " c " " 123 " " camelC " " camelC1 "
Camel Casing	Is Not Set	" caSing " " caSing123 " " \$&% cAsing " " @ casing " " casing " " casinG1 casing2 " " casing1 Casing2 Casing3 " " cASING " " Casing " " camel1c " " c " " 123 " " camel_ Casing " " camelC " " camelC1 "	Should identify the objects " casing1 Casing2 Casing3 " " cASING " " Casing " " camel_ Casing "

## 2.3.2 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 V16 UPD1 is open
- Testsuite 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 "Styleguide\_1":


1. Under the "Project tree" area, select "Project 1 > Test Suite > Styleguide".
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 "Does not contains"
  - Value: One or more characters within the range 1-128Select the "Violation condition" as "Contains" or "Does not contains" according to your requirements
5. Insert any "Value"
6. To run the rule set check. click 

## Example

Violation Condition	String Specified	Object Name	Test Result
Contains	"check"	"To_Check" "To_Ch_ec_k" "To_check"	Identifies the objects "To_Check" "To_check"
Does not conatins	"check"	"To_Check" "To_Ch_ec_k" "To_check"	Identifies the objects "To_Ch_ec_k"
Contains	"name"	" name123" "name123" "name 123" " _ name123"	Identifies the objects " name123" " _ name123"
Does not conatins	" name"	" name123" "name123" "name 123" " _ name123"	Identifies the objects "name123" "name 123"

## 2.3.3 Metadata

### Introduction

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

### Prerequisites

- TIA Portal V16 UPD1 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.

## Procedure

To run the meta data rule for "Rule set\_1":

1. Under the "Project tree" area, select "Project 1 > TestSuite > Styleguide".
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 "BlockmetaRule\_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 .



## 2.3.4 Name Length

### Introduction

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

### Prerequisites

- TIA Portal V16 UPD1 is open
- Testsuite 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 > Styleguide".
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 2-127Select 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 PNV	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.

### 2.3.5 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 V16 UPD1 is open
- Testsuite 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 > Styleguide".
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. To run the rule set check, click .

## Example

Violation Condition	String Specified	Object Name	Test Result
Starts with	"Pre"	"precondition" "Precondition" "Precondition" "_Precondition" "@Precondition"	Identifies the objects "precondition" "Precondition"
Starts with	" Pre"	"precondition" "Precondition" "Precondition" "_Precondition" "@Precondition" "name Precondition"	Identifies the objects " Precondition"
Ends with	"condition"	"preconditional" "Precondition" " Preconditional" "_Precondition" "@Precondition"	Identifies the objects "Precondition" "_Precondition" "@Precondition"
Ends with	"motor"	"start_motor" "start_motor123" "start mot0_r1233" "Start Motor"	Identifies the objects "start_motor" "Start Motor"

## 2.4 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.Interface (In,Out,InOut) CodeBlock.Interface.Input CodeBlock.Interface.Output CodeBlock.Interface.InOut CodeBlock.Interface.Static CodeBlock.Interface.Temp CodeBlock.Interface.Constant GlobalDataBlock GlobalDataBlock.Static InstanceDataBlock PLCDataType PLCDataType.Member PLCTagTable PLCTagTable.Tags PLCTagTable.Constants
Metadata	CodeBlock (FB,FC,OB) CodeBlock.Interface (In,Out,InOut) CodeBlock.Interface.Input CodeBlock.Interface.Output CodeBlock.Interface.InOut CodeBlock.Interface.Static GlobalDataBlock GlobalDataBlock.Static InstanceDataBlock PLCDataType PLCDataType.Member PLCTagTable.Tags

## 2.5 Display of test results

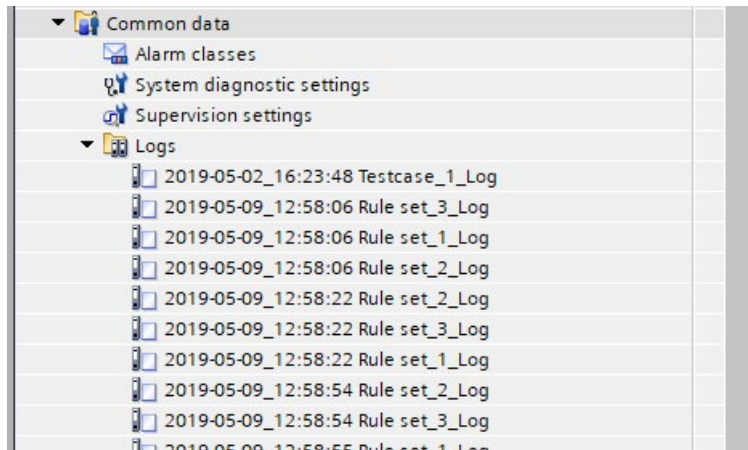
### Test Result display and Log File Creation for Rule set Execution

Execution of a rule set will generate violations, and these messages are displayed in the "Test Results" tab page in the inspector frame. Additionally, 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.

General		Cross-references	Compile	Test Results			
			Show all messages				
Rule set execution finished (errors : 0 ; warnings : 0)							
	Path	Description	Go to	?	Errors	Warnings	Time
	▼ Styleguide				0	0	2:03:15 PM
	▼ Rule set_1				0	0	2:03:16 PM
	CasingRule_1	Violated/Validated CodeBlock (FB,FC,OB) : 0/1, Violation Percentage - 0%					2:03:17 PM
	▼ Elements without violati..				0	0	2:03:17 PM
		PLC_1 - Main : Rule set_1 executed successfully.					2:03:17 PM
	ContainsRule_1	Violated/Validated CodeBlock (FB,FC,OB) : 0/1, Violation Percentage - 0%					2:03:17 PM
	▼ Elements without violati..				0	0	2:03:17 PM
		PLC_1 - Main : Rule set_1 executed successfully.					2:03:17 PM
	NameLengthRule_1	Violated/Validated CodeBlock (FB,FC,OB) : 0/1, Violation Percentage - 0%					2:03:17 PM
	▼ Elements without violati..				0	0	2:03:17 PM
		PLC_1 - Main : Rule set_1 executed successfully.					2:03:17 PM

## 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 PNV 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.
-

## 2.6 Additional Features

### Introduction

Styleguide checker also supports the following features:

1. Master copy support
2. Import/Export of rules from rule set editor

### Master Copy Support

Master copy support is used to transmit rule sets defined in styleguide 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 of 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.

# Application Test

## 3.1 Introduction

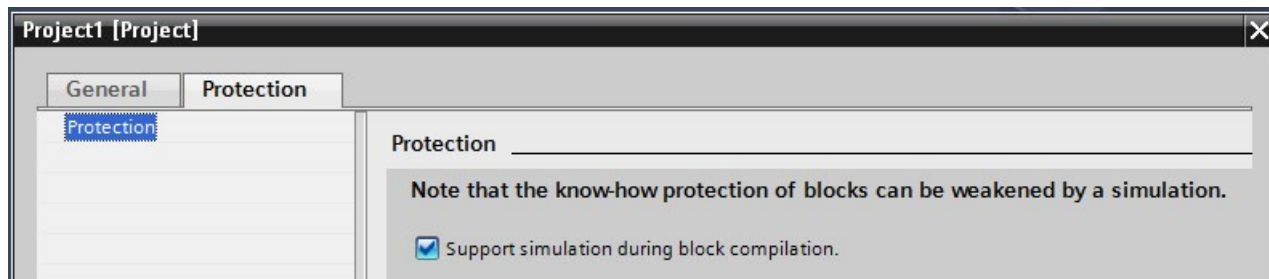
### Introduction

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. It enables the user to:

- 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.

To run an application test, TIA Portal Test Suite must be installed together with SIMATIC S7-PLCSIM Advanced V3.0 Update 1 on the same PC. Hence, the following restrictions of PLCSIM Advanced must be considered:

- Computer with a minimum of 2 processors and 6 GB RAM
- S7-1500 Standard/Compact/Technology/Failsafe controller, ET200SP controller
- Supported firmware  $\geq$  V1.8
- Simulation support must be enabled in the project properties and for all library types/KHP blocks



For more information, refer to PLCSIM Adv. manual



## 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.
- 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 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

## Benefits

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

## 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 V16 Update 1 installed.
- SIMATIC S7-PLCSIM Advanced V3.0 Update1 is installed
- Optional Package TIA Portal Test Suite Advanced V16 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 rule set "test case\_1" is added to the PNV.

---

#### Note

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

## Deleting a Test Case

You can delete a test case from the PNV and deleting will remove the same from the project PNV 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 testcase(s) to be deleted. You can choose "Yes" or "No" based on your requirements.

---

## Renaming a testcase

You can rename a testcase in the PNV. After you rename testcase(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 > TestSuite > 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 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 PNV

You can copy test case(s) from the PNV. This results in addition of the added test case to the respective project PNV under the "Application test" subfolder with the name similar to PNV 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 PNV.
  - 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 within the same project, and not across different projects
-

## 3.3 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 : "InstDB".boolVar := TRUE;
"motor Start" : "InstDB".boolVar := TRUE;
END_VAR
```

Where, `motorStart` is the alias name for the `boolVar` which is present in "InstDB" of selected PLC in the scope.

---

#### 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

6		<Add new>			
7	Static				
8	gain	Real	0.0	Retain	
9	scaledValue	Real	0.0	Retain	
10	setPointValue	Real	0.0	Retain	
11	error	Bool	false	Retain	
12	<Add new>				

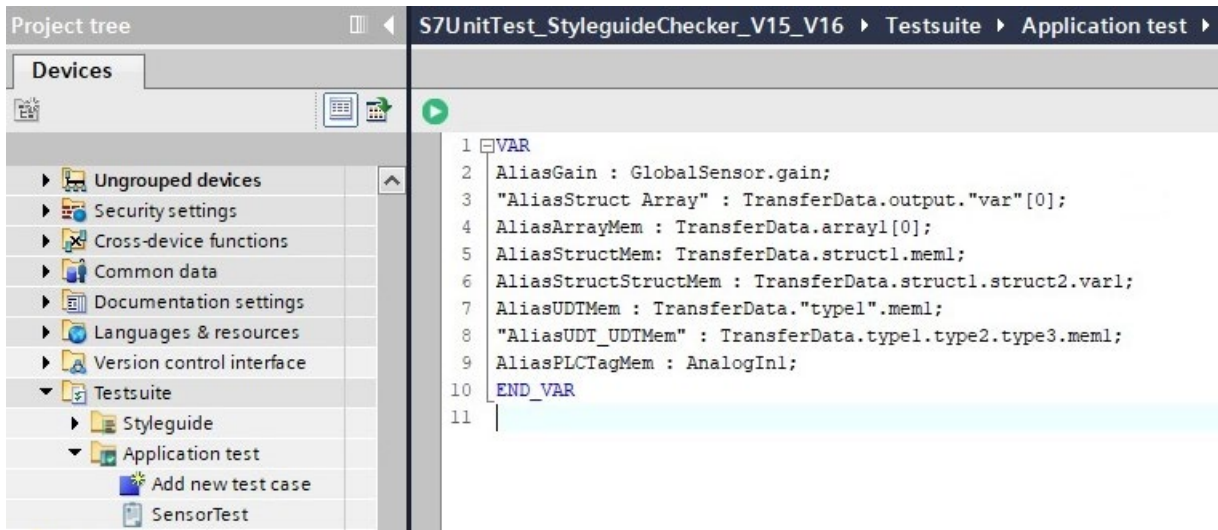
## Transfer Data

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	<input type="checkbox"/>
2	AnalogIn2	Word	%IW2	<input type="checkbox"/>
3	<Add new>			<input type="checkbox"/>

### Test Case Editor Example



## 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 PLC variable/PLC tag/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 depending 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 V3 manual on page 252 (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.

---



## Assert Statements

The ASSERT statements, compare the actual value of the PLC variables/tags with the expected value specified by the user.

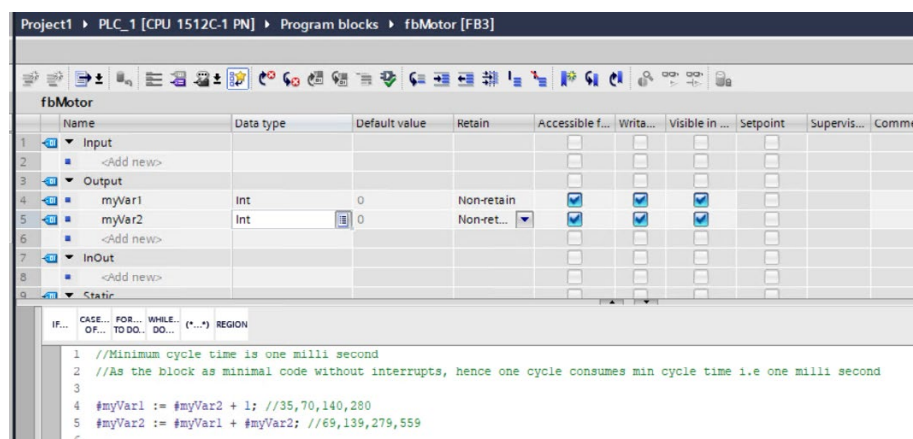
Syntax: Assert.<\_Condition\_>(<actual\_value>,<expected value>);

### 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.Equal 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.Equal validates whether the first parameter is greater than the expected value	Integers /Binary numbers /Floating point numbers/Char/Time
Assert.GreaterThanOrEqual()	actual_value, expected_value	Assert.Equal 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.Equal 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.Equal validates whether the first parameter is less than or equal to expected value	Integers /Binary numbers /Floating point numbers/Char/Time except BOOL datatype

The below example shows the working of instructions in a test case:



Test case for the above PLC program data:

```
VAR
cVar1 : fbMotor_DB.myVar1;
tcVar2 : fbMotor_DB.myVar2;
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
```

### 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 editor 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.

## 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	
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)	-2_147_483_647 to +2_147_483_647.	0	16#0000_0000 to 16#FFFF_FFFF
	Unsigned integers (decimal system)	0 to 4_294_967_295		
	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_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#0 to 16#7F	
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_11	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

Data Types	Datatype formats	Value Range	Examples	Value format displayed in test results
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_11 11_1111_1111_1111_1 111	2#0000_0000_0000_00 01_1110_1011_0101_1 110	+2_147_483_647
	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_11 11	2#1111_1111_0000_11 11	
	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_11 11_1111_1111_1111_1 111	2#1111_0000_1111_00 00_1111_0000_1111_0 000	
	Hexadecimal numbers	16#0 to 16#FFFF_FFFF	16#F0F0_F0F0	
Lint	Signed integers (decimal system)	- 9_223_372_036_854_7 75_808 to +9_223_372_036_854_ 775_807	+154_325_790_816_15 9	- 9_223_372_036_8 54_775_808 to +9_223_372_036_ 854_775_807
	Binary numbers (only positive)	2#0 to 2#0111_1111_1111_11 11_1111_1111_1111_1 111_ 1111_1111_1111_1111 _1111_1111_1111_111 1	2#0000_0000_0000_00 00_1000_1100_0101_1 011_1100 _0101_1111_0000_111 1_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_11 11_1111_1111_1111_1 111 _1111_1111_1111_111 1_1111_1111_1111_11 11	2#0000_0000_0000_00 00_1000_1100_0101_1 011_1100 _0101_1111_0000_111 1_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;	
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;	

### 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_648 ms to T#+24d_20h_31m_23s_ 647ms	T#10d_20h_30m_20s_6 30ms, TIME#10d_20h_30m_2 0s_630ms	T#- 24d_20h_31m_23s _648ms to T#+24d_20h_31m _23s_647ms

### 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'

## 3.6 Test Case Execution

### Introduction

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	Version V1.8, V2.0, V2.1, V2.5,V2.6,V2.8	
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 1510SP F-1 PN CPU 1512SP F-1 PN
Technology CPUs	CPU 1511T-1 PN CPU 1515T-2 PN CPU 1516T-3 PN/DP CPU 1517T-3 PN/DP	CPU 1511TF-1 PN CPU 1515TF-2 PN CPU 1516TF-3 PN/DP CPU 1517TF-3 PN/DP

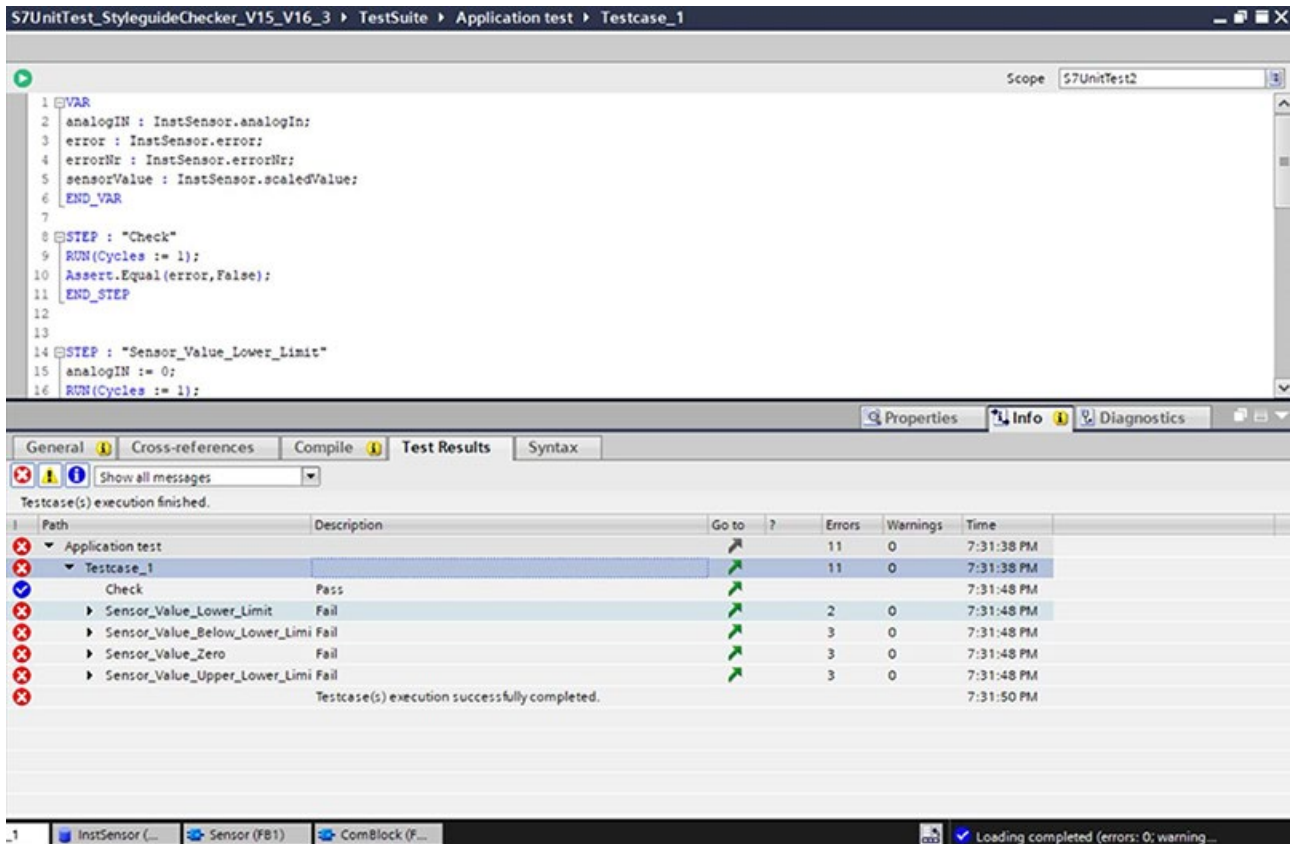
The test execution can be started in two different ways, either from the current opened test case editor or with a context menu entry on the test case in the project tree.

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.

## 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:

- PLCSIM Advance V3.0 Update 1 should be installed in the target.
- User should select the valid 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 PLC should be error free.

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 5,00,000 tags)

For further assistance ,please contact Siemens Customer Support.

## 3.8 Additional Features

### Introduction

Application supports the following features:

1. Master copy support

### 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.