



Siemens
Industry
Online
Support

FAQ

SIMATIC Modular Application Creator Test Environment

SIMATIC Modular Application Creator

SIEMENS

This entry is from the Siemens Industry Online Support. The general terms of use (http://www.siemens.com/terms_of_use) apply.

**Security-
information**

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit <https://www.siemens.com/industrialsecurity>.
Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under <https://www.siemens.com/cert>.

Table of contents

- 1. Overview3**
 - 1.1. Functionality 3
 - 1.2. Test Categories..... 3
 - 1.3. Requirements 3
- 2. Set-up4**
 - 2.1. Setup of the Tests project 4
 - 2.2. Set up Visual Studio Environment..... 6
- 3. Working with the Test project.....7**
 - 3.1. Project Structure..... 7
 - 3.2. Launching a Test 8
 - 3.3. Generation Tests 8
 - 3.4. Function Tests 10
 - 3.5. Updating the TIA Portal project..... 11
- 4. Appendix12**
 - 4.1. Application support 12
 - 4.2. Links and literature 12
 - 4.3. Change documentation 12

1. Overview

1.1. Functionality

The SIMATIC Modular Application Creator Test Environment can be used for a quicker and easier development time. It provides the infrastructure for testing small functions or the whole generation process without the need of starting the SIMATIC Modular Application Creator. The test can be started and debugged directly in Visual Studio without referencing the SIMATIC Modular Application Creator.

1.2. Test Categories

The Test Environment is designed for two different test categories: the generation tests and the function tests.

The generation test can be used for testing/debugging the whole generation process. This can be used as a first starting point for developing the equipment module. Further information can be found in chapter 3.3.

The function test describes the test of single generation functions. With this opportunity single generation functions can be tested/debugged without the need of starting the whole generation process. For preconditions of the function test have a look at chapter 3.4.

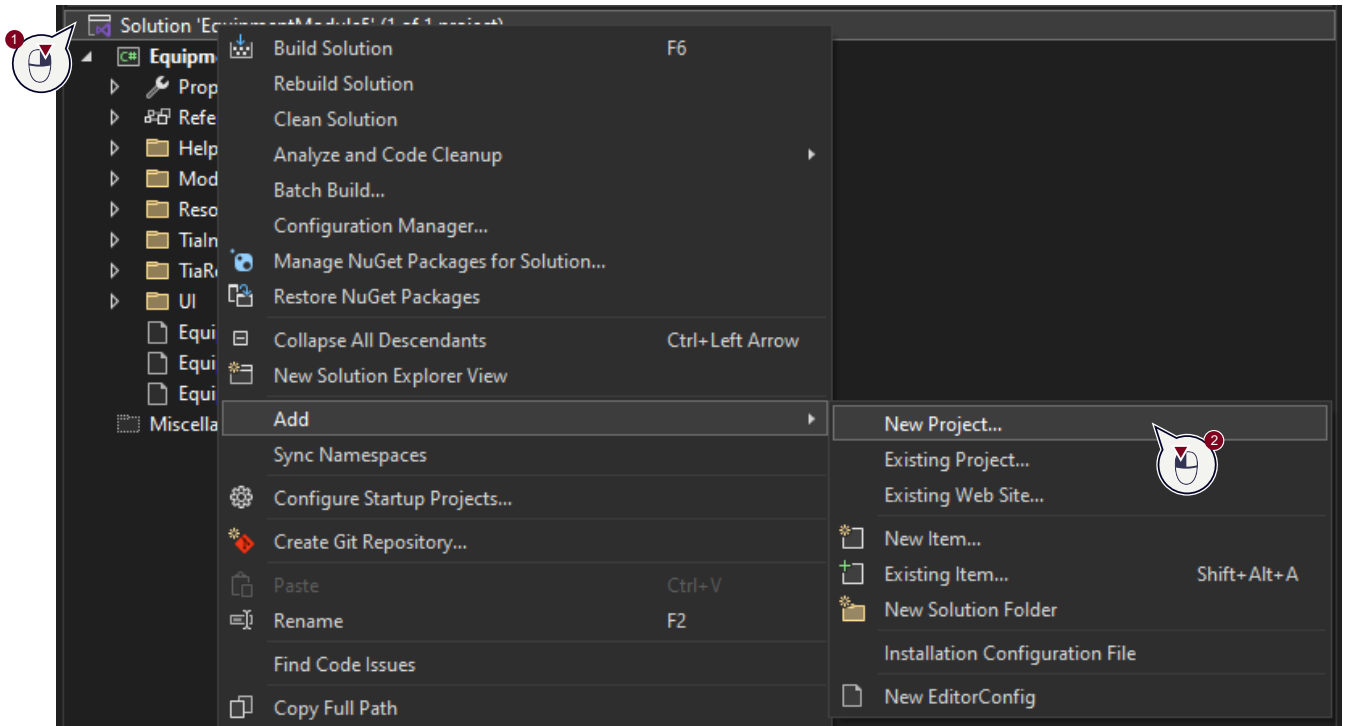
1.3. Requirements

- SIMATIC Modular Application Creator Version \geq V19.2
- TIA Portal \geq V19 with installed SINAMICS Startdrive \geq V19
- Added user to the openness user group. For detailed information look at SIOS article [109773802](#) in chapter Installation: Add user to the user group "Siemens TIA Openness"

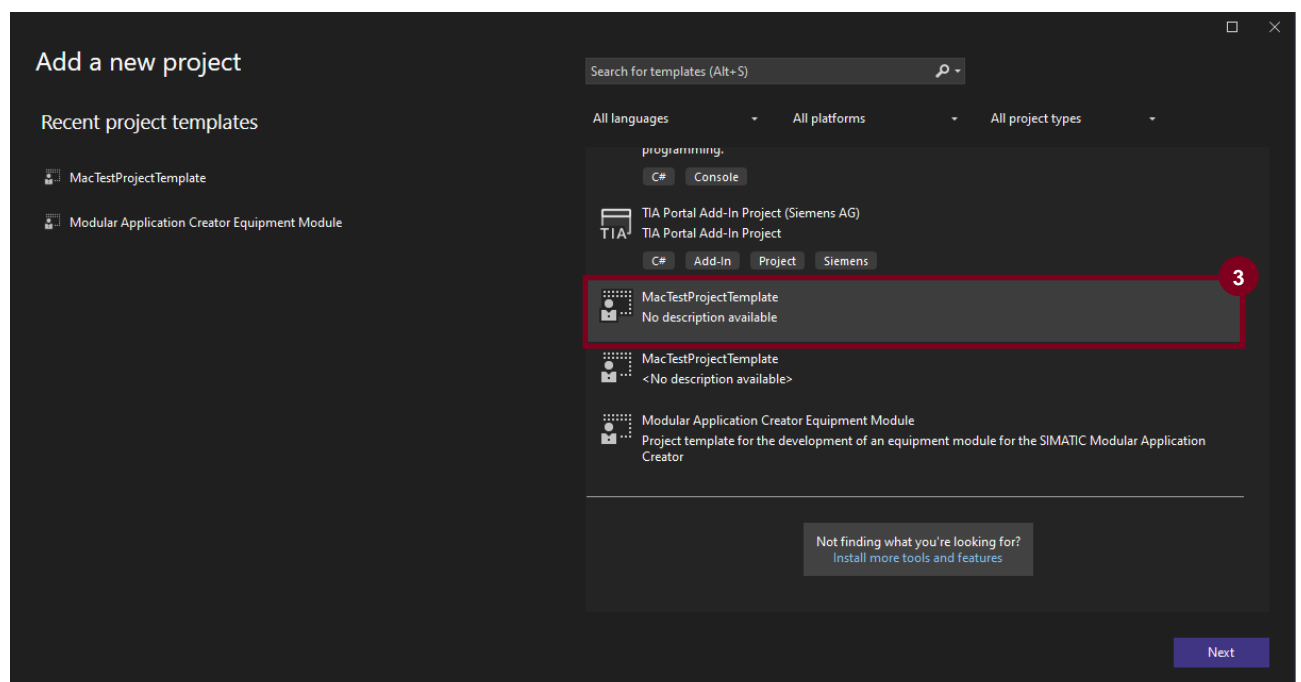
2. Set-up

2.1. Setup of the Tests project

The project for the test environment will be automatically included when generating a new module with the SIMATIC Modular Application Creator Module Builder. It can also be added to existing projects by right clicking the solution (1) and selecting "Add" and then clicking on "New Project..." (2).



Select the "MacTestProjectTemplate" (3) and set the name of the project: "ModuleName".Tests (4). Now click "Create" (5).



Configure your new project

MacTestProjectTemplate

Project name

EquipmentModule.Tests

4

Location

D:\MacProjects\EquipmentModuleSolution\

...

Project will be created in "D:\MacProjects\EquipmentModuleSolution\EquipmentModule.Tests\"

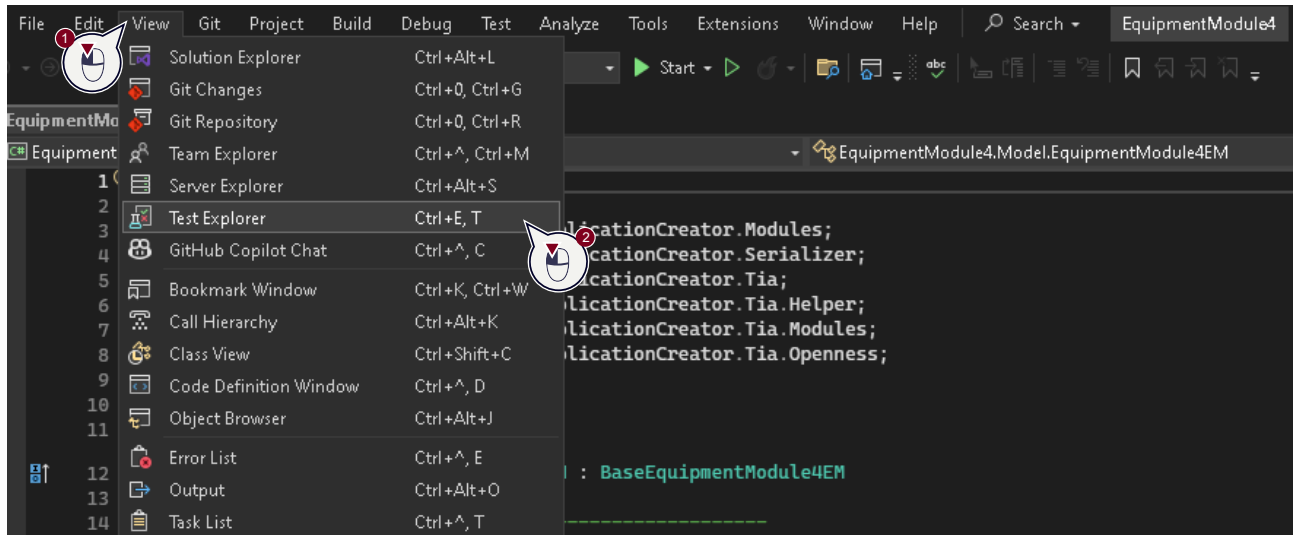
Back

Create

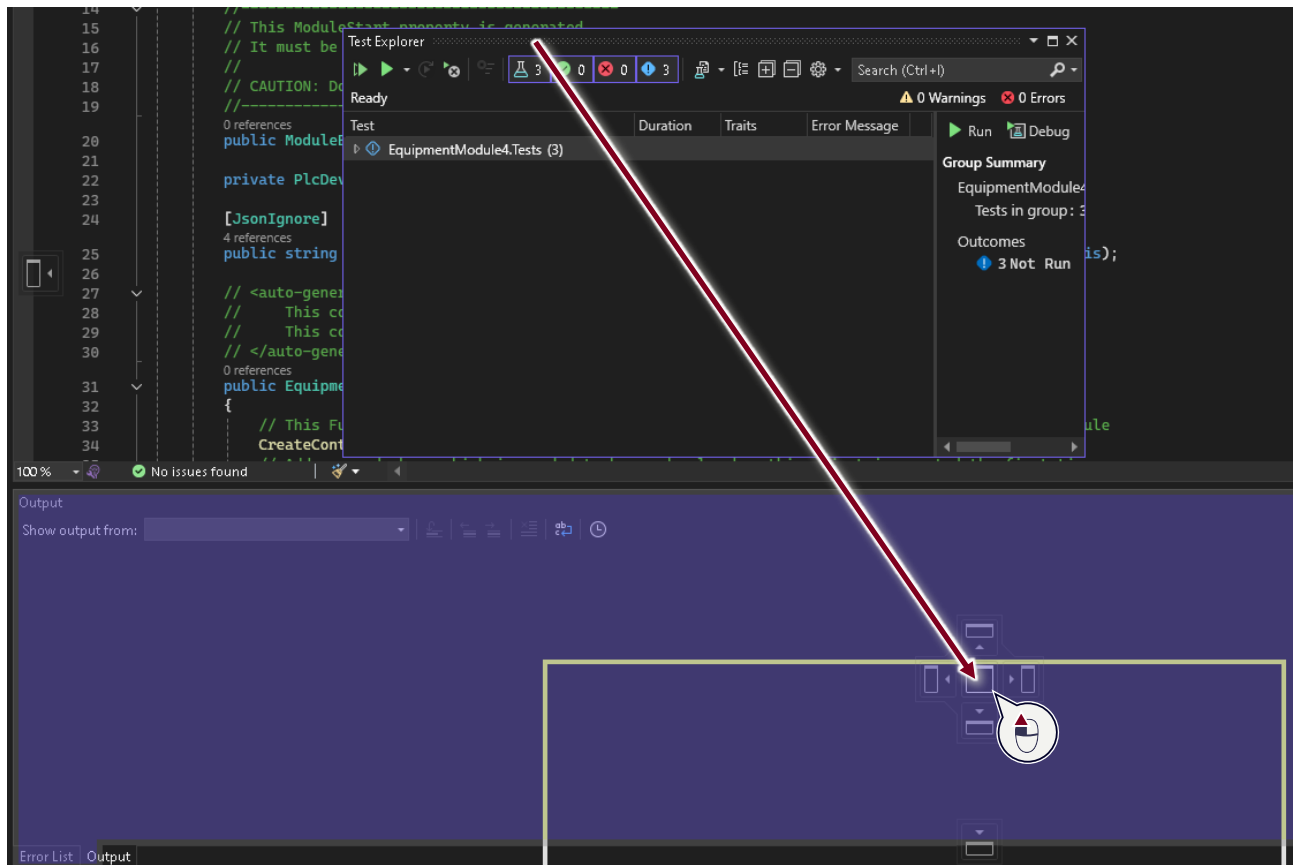
5

2.2. Set up Visual Studio Environment

For executing the tests, the Visual Studio Test Explorer needs to be activated. You can show the test explorer by clicking on the “View” Tab in the taskbar (1) and then on “Test Explorer” (2).



Afterwards the test explorer window appears and can be placed in the window bar next to the output.



For further information about the Test Explorer visit <https://learn.microsoft.com/en-us/visualstudio/test/run-unit-tests-with-test-explorer?view=vs-2022>.

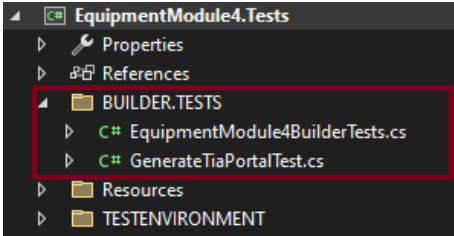
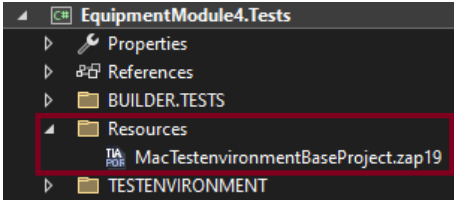
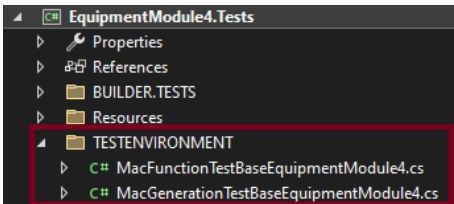
(Other test explorers like dotCover from JetBrains could also be used but are not necessary.)

3. Working with the Test project

The test project contains the infrastructure to test the equipment module with a dedicated test project and built in tests in visual studio. For more general information about how to create a unit test and setup a test project see <https://learn.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2022>.

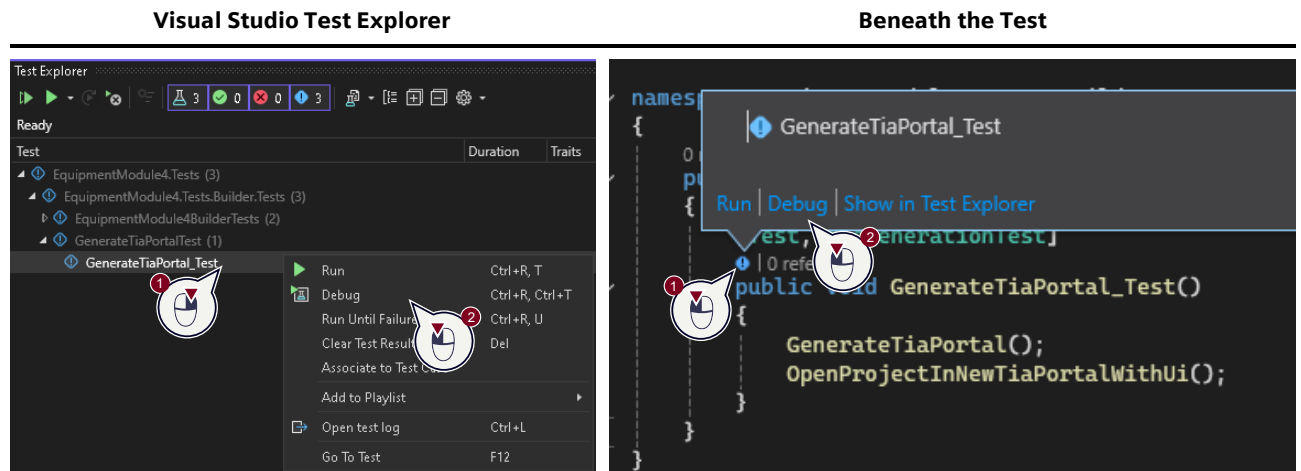
3.1. Project Structure

The project is structured in three different folders:

Project Folder	Description
	<p>The BUILDER.TESTS folder contains the classes containing the real tests which can be executed. The class <i>GenerateTiaPortalTest</i> contains the generation tests and the class <i>...BuilderTests</i> contains examples of function tests. The function tests depend on the module architecture and have some preconditions. For further information look at chapter 3.4.</p>
	<p>The Resources folder contains all necessary resources for the Tests. The zap file from a TIA Portal project is used as a base project for generation of the test. For more information about updating the TIA Portal project look at the chapter 3.5.</p>
	<p>The TESTENVIRONMENT folder contains all base classes needed for testing. It contains a base class for the generation tests ("<i>MacGenerationTestBase...</i>") and one class for the single function tests ("<i>MacFunctionTestBase...</i>").</p>

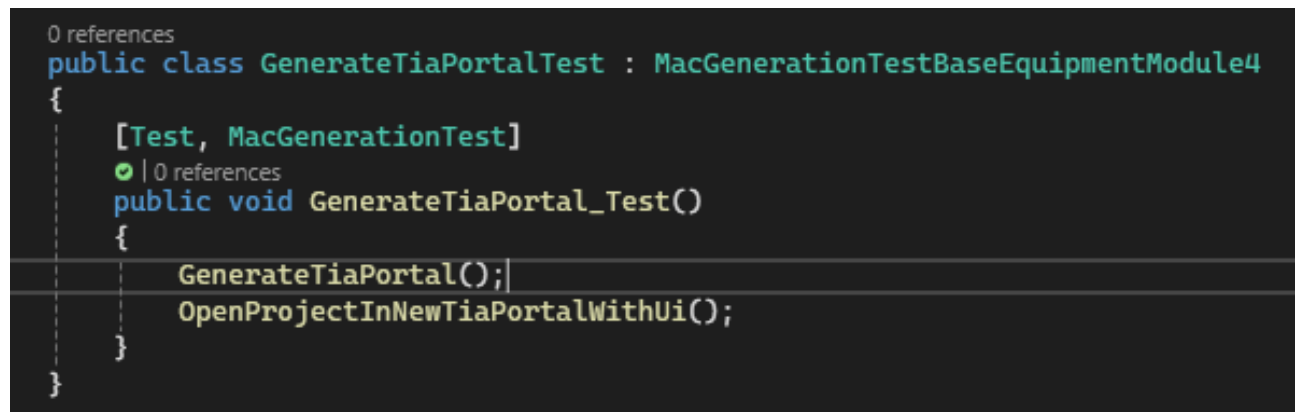
3.2. Launching a Test

The Tests can be launched either via the **Visual Studio Test Explorer** or directly **beneath the Test** itself. The Test could be executed in Debug mode or just started without any debug options. After the test the result can be seen in the Test Explorer and the generated TIA Portal project will be opened automatically.

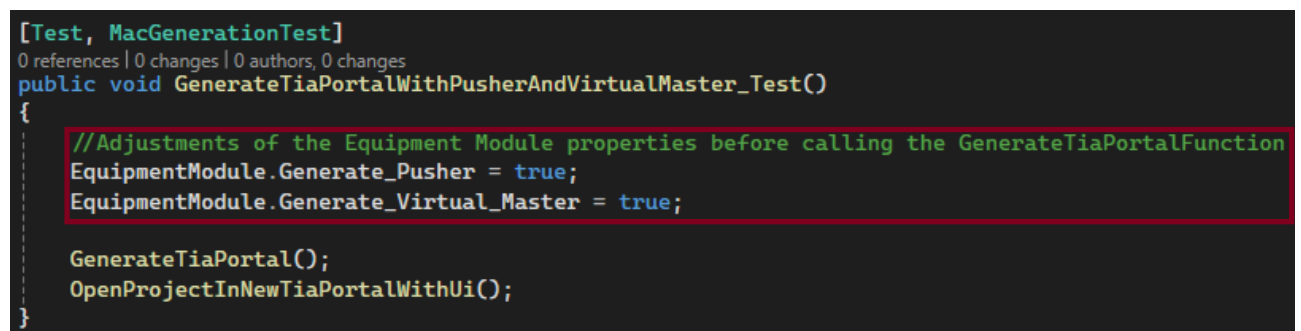


3.3. Generation Tests

As described in chapter 3.1 the Generation Test can be found in the folder Builder.Tests. The test itself calls two functions, the `GenerateTiaPortal()` and the `OpenProjectInNewTiaPortalWithUi();`.



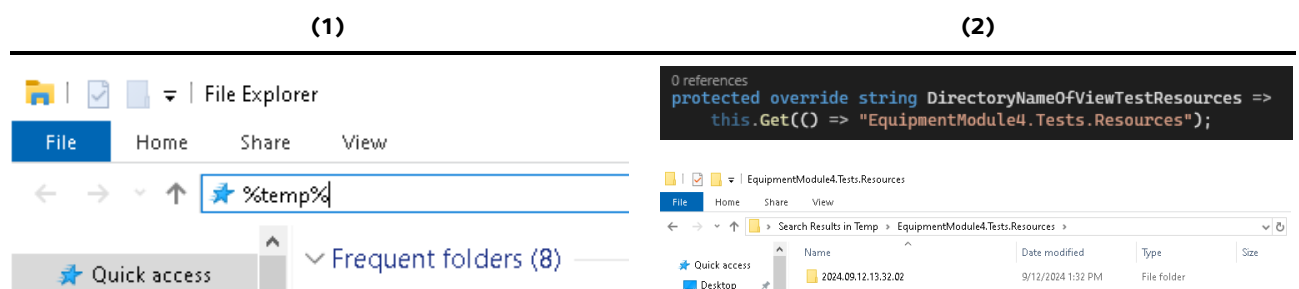
If you want to change the configuration of the Equipment Module (e.g. to simulate a specific UI input) before generating the TIA Portal project in a specific scenario/test you can copy the test with default settings and adjust the class EquipmentModule before calling the GenerateTiaPortal function:



For initialization processes that have to be done before every Generate Test you could adjust the *InitializeModuleClassesForTesting()* function in the *MacGenerationTestBase...* class (see chapter 3.1 for localization).

```
2 references | 0 changes | 0 authors, 0 changes
public class MacGenerationTestBaseUseCaseBasedDoku : MacGenerationTestBase<UseCaseBasedDokuEM>
{
    [TearDown]
    0 references | 0 changes | 0 authors, 0 changes
    public override void Teardown()
    {
    }
    0 references | 0 changes | 0 authors, 0 changes
    protected override void InitializeModuleClassesForTesting()
    {
        //If properties are relevant for each generation process place them here
        EquipmentModule.Generate_Pusher = true;
        EquipmentModule.Generate_Virtual_Master = true;
    }
}
```

All generated projects are saved under the local user's temp directory. This can be reached via typing %temp% in the file explorer (1). The name of the underlying folder is defined via the variable *DirectoryNameOfViewTestResources* in the *MacGenerationTestBase...* class (2).



When another PLC from the project shall be selected for the generation tests in general the properties *PlcName*(1), *PlcCPUName*(2) and *OrderNumberPLC* can be adjusted.

```
0 references
protected override string PlcName => this.Get(() => "S7-1500/ET200MP station_1");
0 references
protected override string PlcCPUName => this.Get(() => "TestenvironmentPLC");
0 references
protected override string OrderNumberPLC => this.Get(() => "");
```

Network overview		Connections	I/O communication	VPN	TeleControl
Device	1	type	Address in subnet	Subnet	Master / IO system
▼ S7-1500/ET200MP station_1		S7-1500/ET200MP stati...			
▶ TestenvironmentPLC	2	CPU 1516T-3 PN/DP			

3.4. Function Tests

As described in chapter 3.1 the Function Tests can be found in the folder `Builder.Tests`. The class contains two example tests how a function test could look like. The structure of the unit tests is depending on the infrastructure of the module itself. The function tests are more advanced than the generation tests and have some preconditions.

For testing single functions, the code should be testable. There are lots of guides on the internet to program testable code. In the context of the SIMATIC Modular Application Creator (when working with static functions like in the Modular Application Creator usecases) the function should have all necessary classes as input parameters. Furthermore, only the needed properties should be given to the function, e.g. the `TiaTemplateContext` shouldn't be part of the input parameters.

The TIA Portal project also needs to be adjusted for the function test. The PLC needs to be in the same condition as in the generation, for example if a specific block needs to be created before the function this must be considered in the TIA Portal project. For updating the TIA Portal project see chapter 3.5.

When another PLC from the project shall be selected for the specific function tests in general the properties `PlcName`(1), `PlcCPUName`(2) and `OrderNumberPLC` can be adjusted. When a specific PLC for a class should be used, copy the properties into the test class and adjust the names (3).

The image displays three components related to configuring function tests:

- Top Code Snippet:** Shows three `protected override string` properties in a test class:
 - `PlcName` (1) returning `"S7-1500/ET200MP station_1"`.
 - `PlcCPUName` (2) returning `"TestenvironmentPLC"`.
 - `OrderNumberPLC` (3) returning `""`.
- Middle TIA Portal Interface:** A screenshot of the 'Network overview' tab. It shows a table with columns: Device, type, Address in subnet, Subnet, and Master / IO system. Two rows are highlighted:
 - Row 1: `S7-1500/ET200MP station_1` (type: `S7-1500/ET200MP stati...`)
 - Row 2: `TestenvironmentPLC` (type: `CPU 1516T-3 PN/DP`)
 Red circles with numbers 1 and 2 point to the 'Device' and 'type' columns respectively.
- Bottom Code Snippet:** Shows a C# class `EquipmentModule4BuilderTests` inheriting from `MacFunctionTestBaseEquipmentModule4`. It contains:
 - A `protected string LanguageSetting` property returning `ProjectOpenness.LanguageSettings.EditingLanguage.Culture.Name`.
 - Three `protected override string` properties:
 - `PlcName` (1) returning `"S7-1500/ET200MP station_2"`.
 - `PlcCPUName` (2) returning `"PLCWithCreatedT0"`.
 - `OrderNumberPLC` (3) returning `""`.
 - A `[Test, MacFunctionTest]` attribute and a `public void GenerateOBMainTest()` method containing test logic.

For most of the SIMATIC Modular Application Creator objects the test framework provides a simulated object. The table describes all accessible properties and functions of the base class specific for the function tests and their use:

Property/Function name	Use of the property
<i>TiaPortal</i>	The TIA Portal used for the test. If no or more than one instances are open, a new TIA Portal will be opened. If one TIA Portal instance is opened the test will try to attach to this instance.
<i>AttachToOpenProject</i>	Set this property in the test to attach to an already opened project. If the value is false a new TIA Portal project for each test will be created/opened. Be careful because this could cause some errors in the unit test when reusing a project.
<i>ProjectOpenness</i>	The openness project object linked with the property <i>ProjectMacSimulated</i>
<i>PlcDeviceOpenness</i>	The openness device object linked with the property <i>PlcDeviceMacSimulated</i>
<i>SoftwareContainer</i>	The Openness object of the software container of the plc device
<i>PlcDeviceMacSimulated</i>	The simulated MAC specific Plc device. This can be used when a function needs the MAC object of a plc device.
<i>ProjectMacSimulated</i>	The simulated MAC specific Project. This can be used when a function needs the MAC object of a project.
<i>GenerateHardware()</i>	Function for generating hardware. This function must be called after adding all the necessary hardware objects in the test.

3.5. Updating the TIA Portal project

There are multiple possibilities to update the TIA Portal project. First the existing TIA Portal project in the *Resources* folder could just be replaced. When the name of the PLC changed the base classes need to be adjusted (described either in chapter 3.3 for generation tests and similar for function tests in chapter 3.4).

When the name of the project file changed the property *NameOfTiaZapFileFromResources* needs to be adjusted:

```
protected override string NameOfTiaZapFileFromResources => this.Get(() => "MacTestenvironmentBaseProject.zap19");
```

Another possibility is to add another project to the resources folder. The project could then be used for example only for the function tests while the generation project stays the same. Then only the *NameOfTiaZapFileFromResources* property from the function tests needs to be adjusted.

The project can also be set on one test class by copying the complete line into the test class:

```
public class GenerateTiaPortalTestClass2 : MacGenerationTestBaseEquipmentModule4
{
    protected override string NameOfTiaZapFileFromResources => this.Get(() => "MacGenerationTestProject_Class2.zap19");

    [Test, MacGenerationTest]
    public void GenerateTiaPortal_Test()
    {
        GenerateTiaPortal();
        OpenProjectInNewTiaPortalWithUi();
    }
}
```

4. Appendix

4.1. Application support

Siemens AG
Digital Factory Division
Factory Automation
Production Machines
DI FA PMA APC
Frauenauracher Str. 80
91056 Erlangen, Germany
mailto: modular.application.creator.industry@siemens.com

4.2. Links and literature

No.	Topic
11	Siemens Industry Online Support https://support.industry.siemens.com
12	SIMATIC Modular Application Creator https://support.industry.siemens.com/cs/document/109762852/simatic-modular-application-creator
13	SINAMICS/SIMATIC – Automation of engineering tasks in machine building https://support.industry.siemens.com/cs/document/109821826/sinamics-simatic-automation-of-engineering-tasks-in-machine-building

4.3. Change documentation

Version	Date	Modification
V1.0	10/2024	First version of the documentation of the SIMATIC Modular Application Creator Test Environment