# LUnit

## Astemes - Anton Sundqvist

*Anton Sundqvist*

# Table of contents

# 1. LUnit Basics

This document walks through the basic workflow using LUnit to test LabVIEW code. If you prefer to watch a video, there is an introduction available on this link.

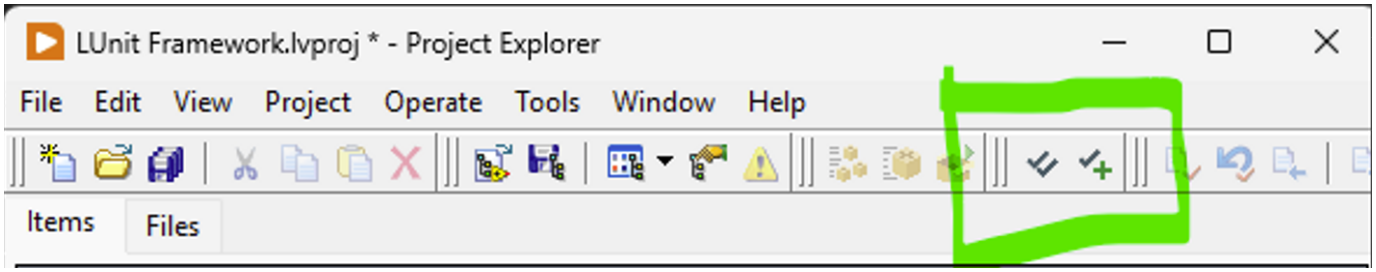
## 1.1 Prerequisites

To follow along with the instructions on this page you will need to have LabVIEW version 2020 or later installed as well as the LUnit unit testing framework.

## 1.2 Creating a Test Case

All tests you write will belong to a test case class. This is implemented as a LabVIEW class, but in order to use it you will not need to know anything about object oriented programming.

To get started, create an empty project and add a test case class to it by clicking the New Test Case button in the toolbar of the LabVIEW project.



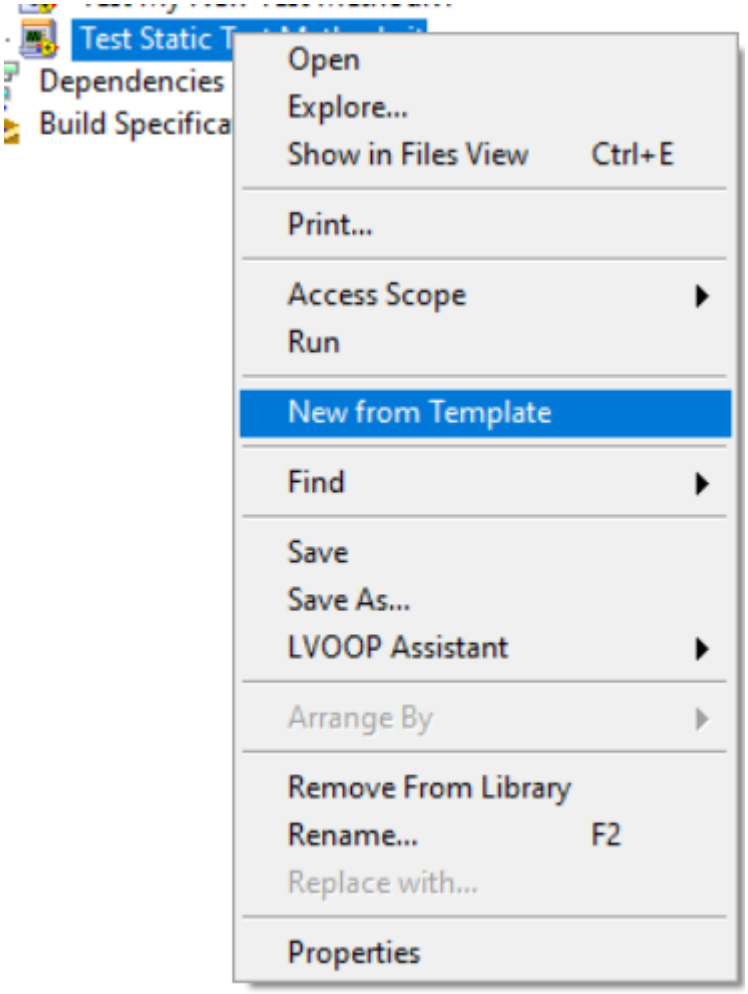

You can also do the same from the `Tools > LUnit > New Test Case...` menu option.


Save the test case in a convenient location. Some like to keep the tests next to the code they are testing, and other keep them in a separate folder called `Tests` or similar. I personally find the later option with a separate top level directory the most convenient. Keep in mind that tests should not be included in builds and there should be no dependencies pointing from your code to the test code.

## 1.3 Adding a Test Method

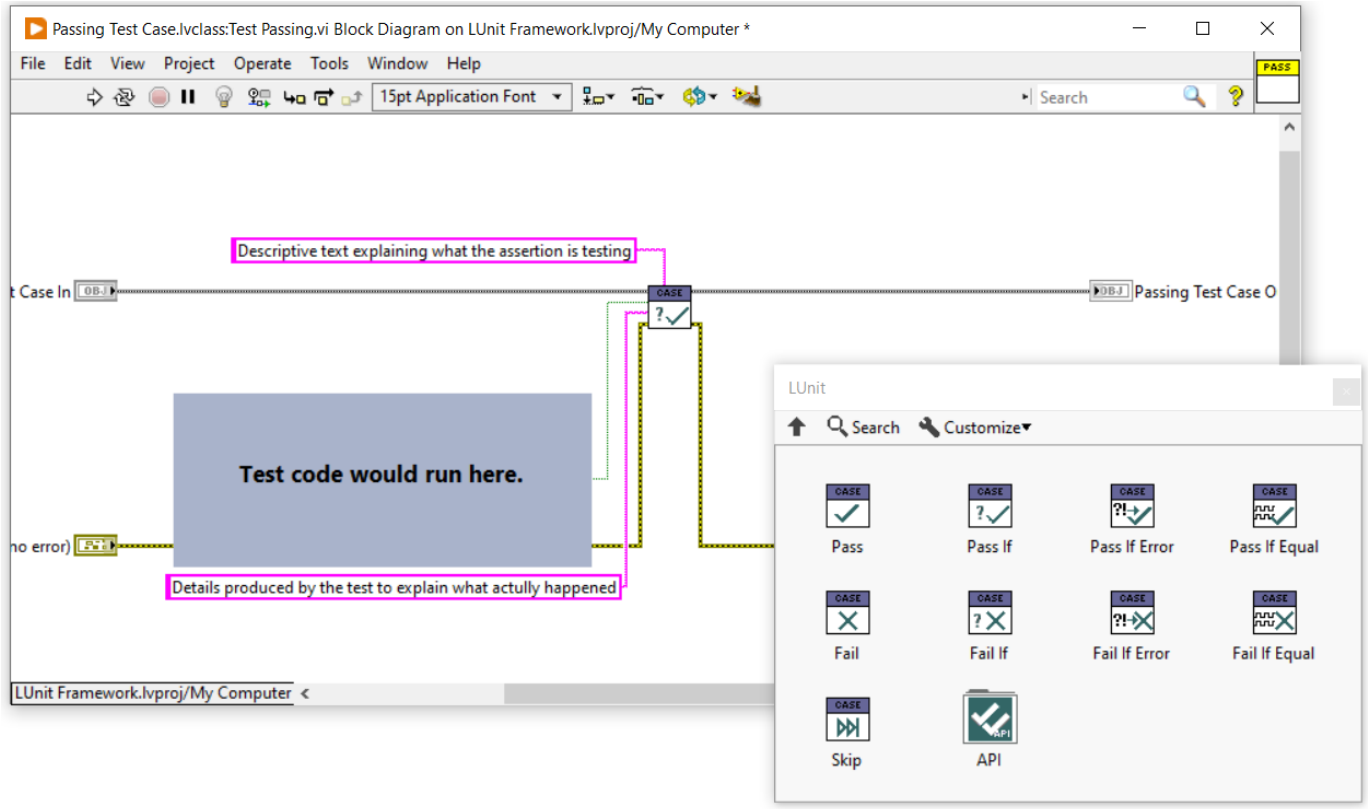
Now you have a test case and may add some test methods to the test case. A test method is a vi belonging to the test case class and will get executed by the framework. The name of the vi **must** start with the four letters "test" (case insensitive). It is **not** recommended to make test vi:s *dynamic dispatch*.

To create a new test method, right-click on the Test Static Test Method.vit and select `New from Template`.




You can create test methods any way you like and you are free to delete the template method. It is important however that the connector pane uses the same pattern of terminals as the template. The error in terminal is optional and never used when running tests by the framework.

You should now make your test method test something useful by implementing the block diagram of the vi. To perform tests you will use the assertions available in the provided palette, or using quick drop.




## 1.4 Using Assertions

The result of each test is determined using assertions. There is a set of assertions to choose from, as shown in the figure above, and the names should be self explanatory. One test method may contain multiple assertions and the result from each assertion will show up in the result view.

One pro-tip is that the `Pass if Equal.vi` assertion also works well for array data types. The result of comparing arrays will show up in the result view as shown below.


Please note that the `Pass if Equal.vi` assertion will fail if either the type or the value does not match.

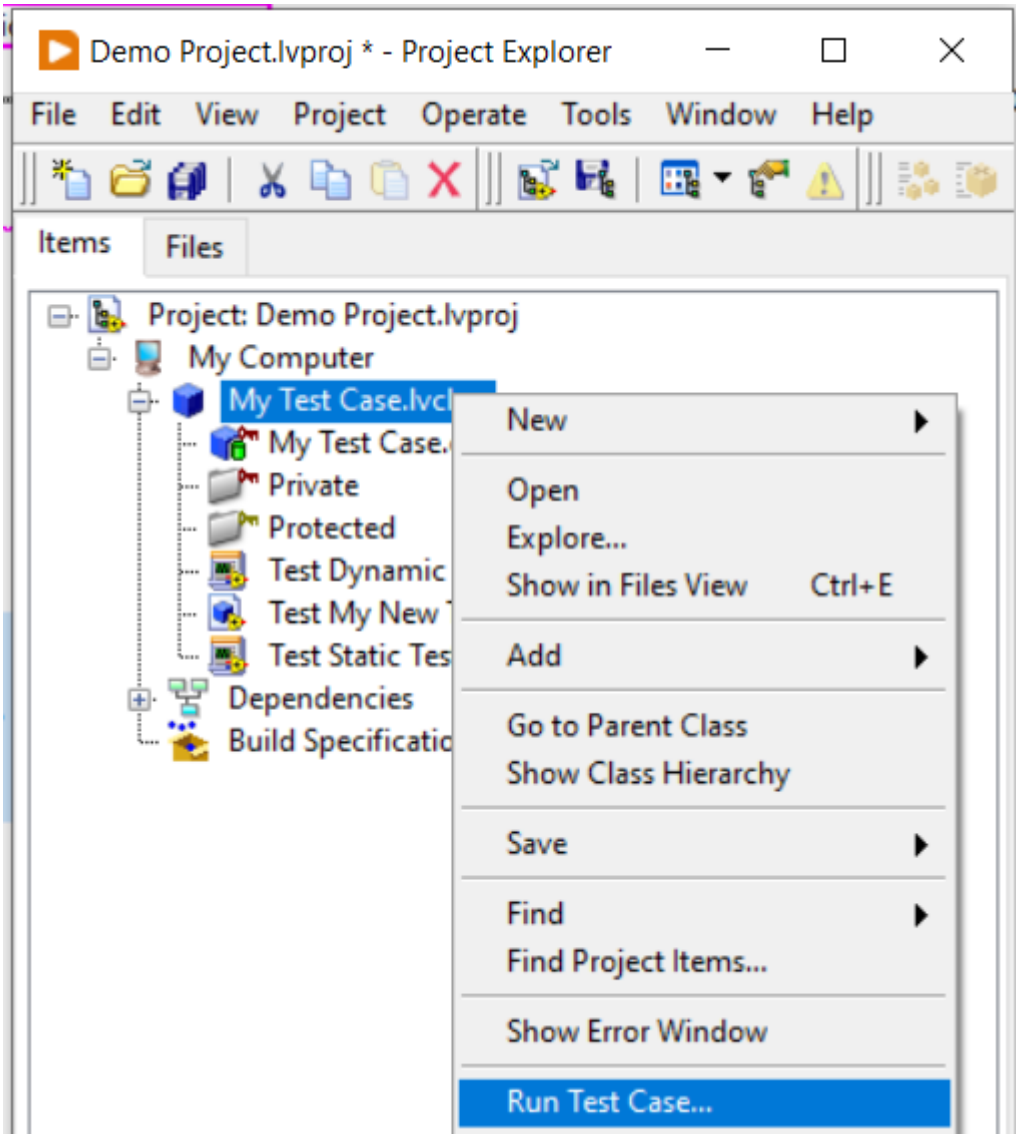
## 1.5 Running the Test Case

You can run a single test vi (using the Run Arrow) and it will run and show the user interface with the results of the test.

Starting from version 1.10 of LUnit, a Quick Drop plugin is available which allows running tests using a Quick Drop shortcut. From Quick Drop press `Ctrl + L` to run all tests within the current project. If you open Quick Drop from a VI and press `Ctrl + Shift + L` LUnit will run all tests from Test Case classes calling this specific VI. This can be very useful for checking if an edit to a VI caused any test to fail. Please note that this feature requires all tests to be loaded into memory,

*i.e.* included in the active project, and it can only check for static links. The last limitation means that the feature will not work for dynamic dispatch VI:s, as their call sites are not statically known.

To run all tests contained in a test case, you can right click it in the project window and select the `Run Test Case...` menu option.




This will open the test execution user interface and run the test case. Alternatively you can also launch the user interface from the tools menu through the `Tools > LUnit > LUnit UI...` menu option. This will open the user interface and show all tests in the current project. As the test is run, the results are also shown as visual icons overlays in the project explorer.