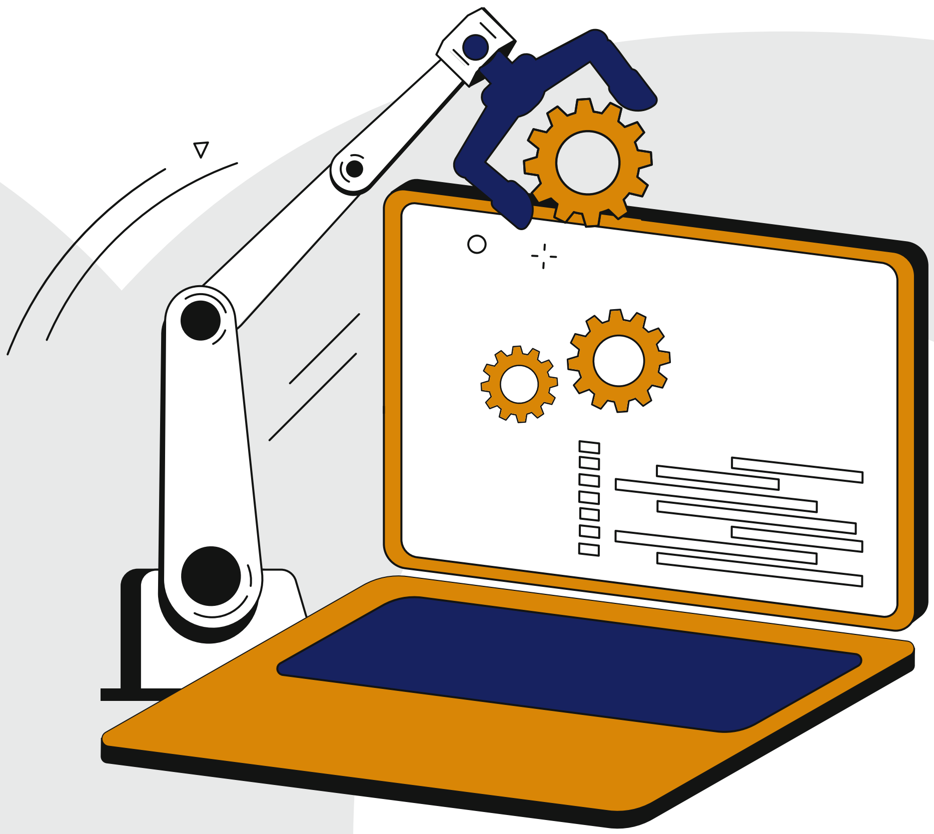
Quick Start for SIGMATEK



Automate Your Code

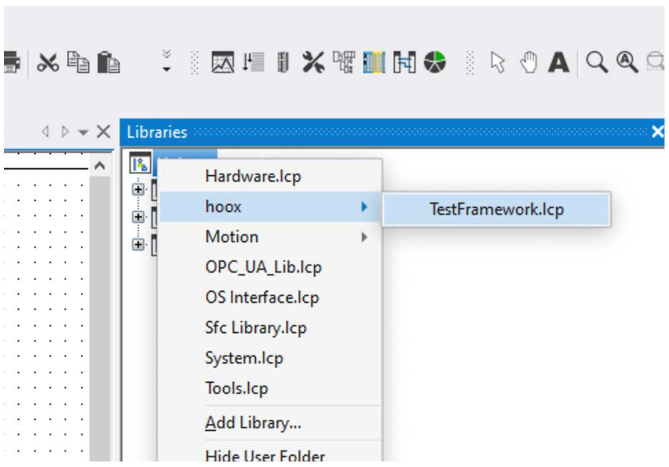
HOOX [SOFTWARE]

HOOX Software
Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

Test Library

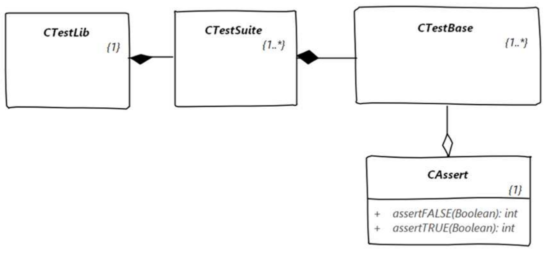
Import Library

Open the library repository management and add the new library.
Then, include the library in the application.



General

The library essentially consists of three parts: CTestlib, CTestsuite, and CTestBase.



Component	Description
`CTestLib`	Manages all configured TestSuites
`CTestSuite`	Manages the associated test cases
`CTestBase`	Base class for a test case
`CStandardTextReport`	Example for exporting the test run
`CReportBase`	Base class for a report that is called after the test run.

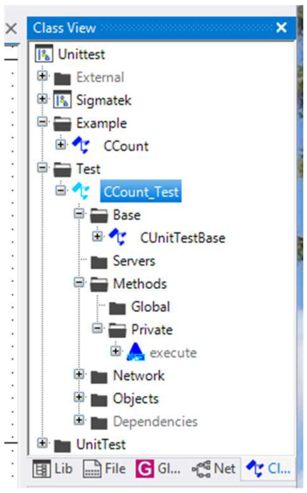
The CTestlib class includes the reports for exporting JUnit and Coverage information by default.

Custom reports can be created by inheriting from CReportsBase.

Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

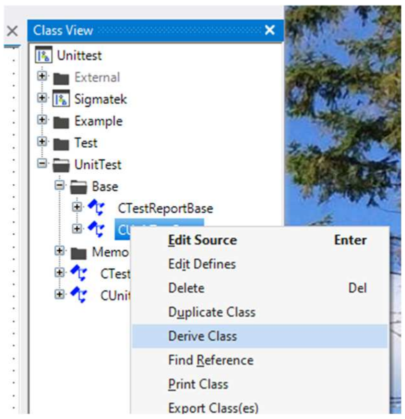
Example

The class CCount is to be tested.

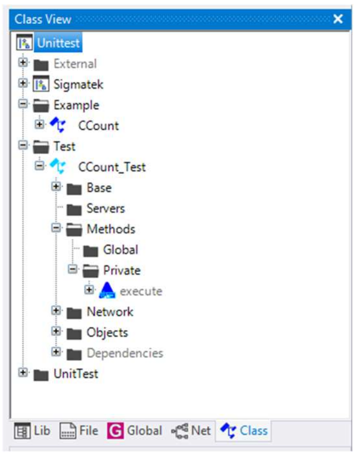


Procedure

1. Create test component
To do this, derive a new (test) class from the CTestBase class.



2. Override necessary methods



To use the test component, the `execute` method (Test Execution) must be overridden.

In the first step, it is sufficient to omit `prepare` and `cleanup`.
Code can be entered in these two methods that should be executed before (`prepare`) or after (`cleanup`) the test case.

Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

3. Implement test

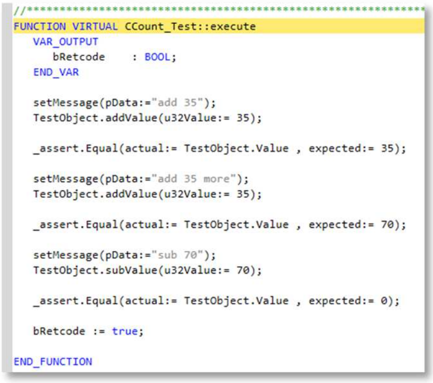
For the classification of the test case ("passed" or "failed"), so-called asserts must be called.
These always consist of an actual and an expected value.

An assert is an assertion that the actual value

corresponds : Assert.assertTrue, or
does not correspond : Assert.assertFalse.
or respectively
corresponds : Assert.Equal, or
does not correspond : Assert.NotEqual.

- Numbers are passed as DINT or REAL
- The same data types must be used for both parameters.

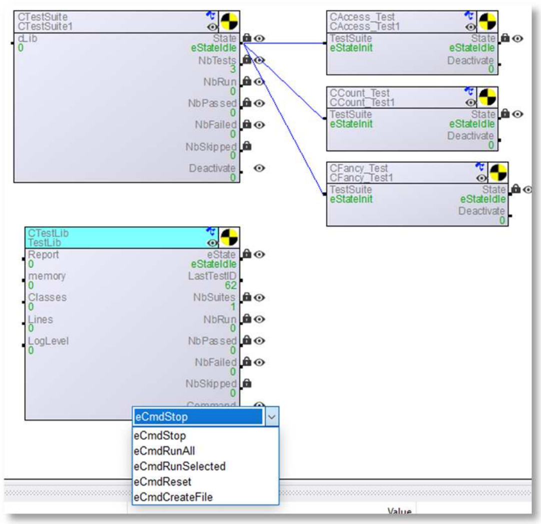
Regarding the example:
The function expects a DINT value and adds or
subtracts this value from the current value of the instance.



```
//*************************************************************
FUNCTION VIRTUAL CCount_Test::execute
   VAR_OUTPUT
      bRetcode     : BOOL;
   END_VAR

   setMessage(pData:="add 35");
   TestObject.addValue(u32Value:= 35);

   _assert.Equal(actual:= TestObject.Value , expected:= 35);

   setMessage(pData:="add 35 more");
   TestObject.addValue(u32Value:= 35);

   _assert.Equal(actual:= TestObject.Value , expected:= 70);

   setMessage(pData:="sub 70");
   TestObject.subValue(u32Value:= 70);

   _assert.Equal(actual:= TestObject.Value , expected:= 0);

   bRetcode := true;

END_FUNCTION
```

Additionally, it is recommended to save a hint via `setMessage(...)` before the call in case the assert fails, in order to get an indication of exactly where the error occurred.

4. Run test



To be able to execute the test cases, the command `eCmdRunAll` must be sent to the Testlib

Patrick Dressel
Steinbühl 1
95233 Helmbrechts
M: +49 170 5260988
patrick@hoox.software

Instantiate Test Components

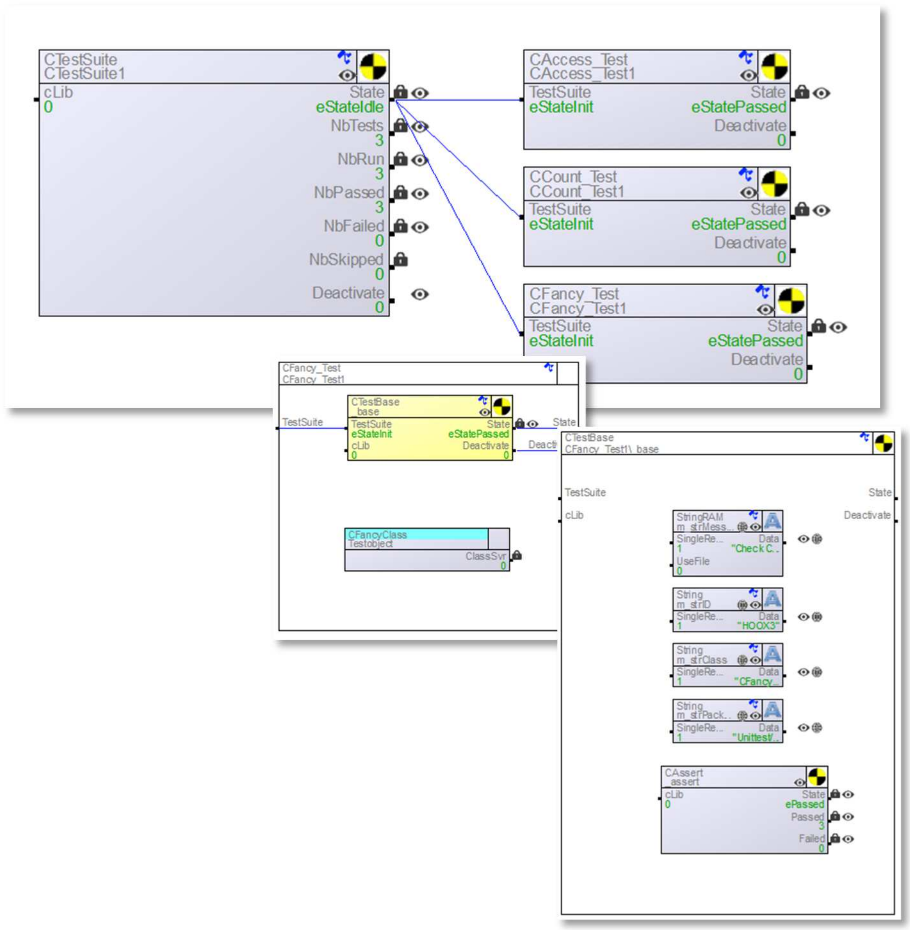
The test components must be instantiated within a global variable list. For execution, a Testsuite that manages multiple tests is needed for a better overview. For later evaluation, further information must be passed to the instance:

Testsuite

Parameter	Description
`m_strName`	Name of the Testsuite

Testfall

Parameter	Beschreibung
`m_strId`	Unique ID of the test
`m_strClass`	Class of the test object
`m_strPackage`	Path to the source code of the contained test cases
`m_strMessage`	Text that can provide information in case of an **error**



HOOX

[SOFTWARE]

www.hoox.software