Verifying Test Cases for Regression Testing

This topic explains how to convert test cases into regression test cases through test case verification.

Sections include:

Generating Regression Tests that Don't Require Verification

If you want to create a snapshot for regression testing (e.g., if you are confident that the code is behaving as expected), use the "**Unit Testing> Generate Regression Base**" built-in Test Configuration.

When this Test Configuration is run, C++test will automatically verify all outcomes.

During subsequent tests C++test will report tasks if it detects changes to the behavior captured in the initial test. Verification is not required.

About Verification

When C++test automatically generates unit tests for code that behaves as expected, the generated test cases form a "functional snapshot": a suite of unit tests that capture the code's current behavior, which is assumed to be correct. This test suite is essentially an executable specification. It can be used to establish a baseline that can be used to identify problems and changes introduced by new/modified code. When your goal is to establish a baseline rather than verify the application's current functionality, you do not need to review the outcomes for these test cases. Moreover, because you want to maintain the integrity of this baseline, you do not want or need to recreate unit tests for this same code. As the application evolves, you can test new and modified code against this baseline to ensure that changes have not impacted or "broken" the previously-verified functionality.

Any test case that you want to use for regression testing should be verified. When a test case is verified, postconditions (which capture the actual value that a variable or class member had during the test execution) are converted to assertions that will be checked in all subsequent tests. After an outcome is verified, the related test will fail if the same value is not achieved.

We recommended verifying user-defined test cases that use postconditions, as well as any automatically-generated test cases that you have reviewed and deemed useful for regression testing. When reviewing automatically-generated test cases, it is helpful to expand the test case node to display the postcondition details, which are automatically added for the function return value and direct member variables for object under test.

Verifying Test Cases

Automated Verification of Test Cases with Unverified Outcomes

Test cases with unverified outcomes (postconditions rather than assertions) can be verified automatically.

To automatically verify one or more test case outcomes that are marked as Unverified Outcomes:

- Right-click the unverified outcome (or a node representing a set of unverified outcomes) in the Quality Tasks view, then choose **Verify Outcome** from the shortcut menu.

All post condition macros (whether added manually or automatically) from selected context(s) will be converted to appropriate assertions. For example:

CPPTEST_POST_CONDITION_INTEGER("{int}_return=", _return) is converted to
CPPTEST_ASSERT_EQUAL(0, _return),

CPPTEST_POST_CONDITION_FLOAT("{float}_return=", _return) is converted to
CPPTEST_ASSERT_DOUBLES_EQUAL(1.000000e+000, _return, 0.00001),

CPPTEST_POST_CONDITION_PTR("{int *}_return=", _return) is converted to
CPPTEST_ASSERT(_return != 0).

Manual Verification

To manually verify test cases:

1. In the project tree, locate the test suite file that C++test generated.
   - By default, automatically-generated test classes are saved in the `tests/autogenerated` directory within the tested project.
   - To check where C++test is saving the test suite files, open the Test Configurations dialog, select the Test Configuration that was used for the test run that generated the tests, then review the value in the **Generation> Test Suite** tab's **Test suite output file and layout** field (see Test suite tab for details).
2. Double-click the project tree node that represents the generated test class. The generated test class file will open in an editor.
3. Use the available macros to convert the postconditions to assertions.
   - See C++test API Documentation for a list of macros.

4. Make any additional modifications needed.
5. Save the modified file.