Reviewing Automatically-Generated Test Cases

This topic explains how to access and explore C++test's automatically-generated test cases.

Sections include:

- Accessing the Test Cases
- Understanding the Test Cases
- Understanding Automatically-Generated Stubs

Accessing the Test Cases

The automatically-generated test cases are saved in test suite files. With the default settings, C++test generates one test suite per tested file. It can also be configured to generate one test suite per function (see Test suite tab for details).

To view the generated 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 or change 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.

Understanding the Test Cases

Test suites contain test cases, using the following architecture:

- **A Test Suite** is a kind of a framework for test cases. It helps in grouping test cases. Each test suite has a class definition that derives from the CppTest_TestSuite class, test case declarations, and definitions of the test case functions.
    - A test suite generated for a "C++" function or class method takes the form of a class and test cases are public methods of this class. If the tested method is a class member, then the test suite class is made a friend to the class that the tested method comes from. This association is very powerful, as it provides the ability to control private members of the tested class from inside a test case.
    - A test suite for C code is used only as a "container" (file) for declaring global functions.
- **A Test Case** is a Test Suite class' public method or a global function (depending on whether it is generated for C or C++) which contains three sections:
    - Arguments/Pre conditions assignments.
    - Tested function calls.
    - Results/Post conditions validations. C++test defines a set of macros for controlling and validating results. These macros can be used for passing messages/assertions from the test executable to the C++test GUI. The available macros are listed at C++test API Documentation.

If more functions are later added to the file under test, then test generation is run on the updated file, additional test cases will be added to the original test suite file.

Context and Include Macros

Every test suite begins with the following two macros:

- `CPPTEST_CONTEXT:` Specifies which file the test suite tests. Only one source file can be specified. If no context is specified, the test suite will be executed whenever the project is executed.
    - When a source file or directory is selected in a project tree before test execution is started, C++test scans all test directories specified in the Test Configuration's test search path. All test suites that match the selected context will be executed.
    - If the entire project is selected, all test suites on the test path will be executed.
    - If a test suite or single test is selected, the CONTEXT macro is used to back-trace to the source file to which this test suite is related. Only the selected test suite(s) will be executed.
- `CPPTEST_TEST_SUITE_INCLUDED_TO`: Indicates that it's an included test suite and specifies which file the test suite source will be included when the test executable is built. When you want to include additional files, use the #include directive.

The files specified by these two macros should always match the name of the file under test. If the name of the file under test changes, the values of these macros must be modified accordingly.

Relative paths can be used in both of these macros. Relative paths must start with "./" or with "../".

Postcondition Macros

The following postcondition macros, are used to report the value that a variable or class member had during the test execution:

- CPPTEST_POST_CONDITION_BOOL(value_string, value)
- CPPTEST_POST_CONDITION_INTEGER(value_string, value)
- CPPTEST_POST_CONDITION_UINTEGER(value_string, value)

- CPPTEST_POST_CONDITION_FLOAT(value_string, value)
- CPPTEST_POST_CONDITION_CSTR(value_string, value)
- CPPTEST_POST_CONDITION_CSTR_N(value_string, value, max_size)
- CPPTEST_POST_CONDITION_MEM_BUFFER(value_string, value, size)
- CPPTEST_POST_CONDITION_PTR(value_string, value)

The results of these macros (all the asserted values reported) are displayed in the Quality Tasks view. Any test case with reported postconditions can be automatically validated for use in regression testing. Validation changes the *_POST_CONDITION_* macros into assertions that will fail if a subsequent test does not produce the expected (validated) value. This is especially useful for automated generation of a regression base for legacy code. For details on validation, see Verifying Test Cases for Regression Testing.

Other Macros

Assertion macros are added to the test case when postconditions are verified.

Additionally, other macros may be added manually, based on your testing needs.

For a list of macros that can be used in test cases, see C++test API Documentation.

Test Functions

Automatically-generated test cases sometimes include special functions (e.g., for accessing minimum of maximum values).

To generate test cases using different numeric values, the C++test runtime library contains a set of functions that returns min/max values of integral and floating point types. The functions are described in the following table. All listed functions can be used in user-defined test cases.

Function	Purpose
`char cpptestLimitsGetMaxChar(void);`	Returns maximum value of char type.
`char cpptestLimitsGetMinChar(void);`	Returns minimum value of char type.
`signed char cpptestLimitsGetMaxSigned Char(void);`	Returns minimum value of char type.
`signed char cpptestLimitsGetMinSigned Char(void);`	Returns minimum value of signed char type.
`unsigned char cpptestLimitsGetMaxUnsigned Char(void);`	Returns maximum value of unsigned char type.
`short cpptestLimitsGetMaxShort(void);`	Returns maximum value of short type.
`short cpptestLimitsGetMinShort(void);`	Returns minimum value of short type.
`unsigned short cpptestLimitsGetMax UnsignedShort(void);`	Returns maximum value of unsigned short type.
`int cpptestLimitsGetMaxInt(void);`	Returns maximum value of int type.
`int cpptestLimitsGetMinInt(void);`	Returns minimum value of int type.
`unsigned int cpptestLimitsGetMaxUnsigned Int(void);`	Returns maximum value of unsigned int type.
`long cpptestLimitsGetMaxLong(void);`	Returns maximum value of long type.
`long cpptestLimitsGetMinLong(void);`	Returns minimum value of long type.
`unsigned long cpptestLimitsGetMaxUnsigned Long(void);`	Returns maximum value of unsigned long type.
`float cpptestLimitsGetMaxPosFloat(void);`	Returns maximum positive value of float type.
`float cpptestLimitsGetMinNegFloat(void);`	Returns minimum negative value of float type.
`float cpptestLimitsGetMaxNegFloat(void);`	Returns maximum negative value of float type.
`float cpptestLimitsGetMinPosFloat(void);`	Returns minimum positive value of float type.
`double cpptestLimitsGetMaxPosDouble(void);`	Returns maximum positive value of double type.
`double cpptestLimitsGetMinNegDouble(void);`	Returns minimum negative value of double type.
`double cpptestLimitsGetMaxNegDouble(void);`	Returns maximum negative value of double type.
`double cpptestLimitsGetMinPosDouble(void);`	Returns minimum positive value of double type.
`long double cpptestLimitsGetMaxPosLong Double(void);`	Returns maximum positive value of long double type.
`long double cpptestLimitsGetMinNegLong Double(void);`	Returns minimum negative value of long double type.
`long double cpptestLimitsGetMaxNegLong Double(void);`	Returns maximum negative value of long double type.

`long double cpptestLimitsGetMinPosLongDou-ble(void);`	Returns maximum positive value of long double type.

Understanding Automatically-Generated Stubs

See Understanding and Customizing Automatically-Generated Stubs.