Reviewing Test Execution Results

This topic covers how to analyze and address C++test's test execution results.

Sections include:

Reviewing Reported Tasks

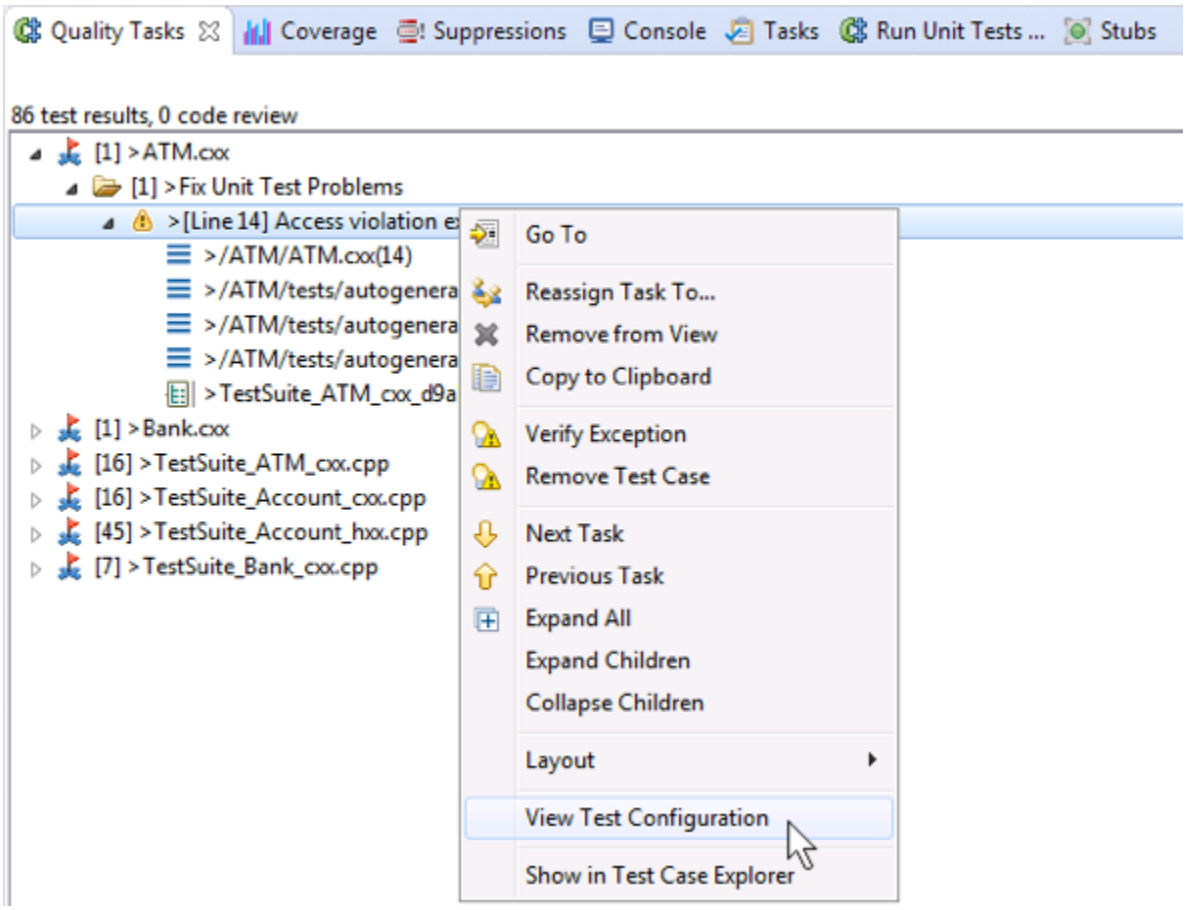
After test execution, C++test generates a prioritized task list organized by error categories and severities. For tests run in the GUI, tasks are organized into the following categories in the Quality Tasks view:

- **Fix Unit Test Problems:** This category contains definite unit test problems—including functional test failures, unexpected exceptions, and timeouts—that need to be addressed.
- **Review Unit Test Outcomes:** This category contains unverified outcomes for test cases that were created during automated test case generation. Unverified outcomes are reported when C++test executes automatically-generated or user-defined test cases with postconditions that have not yet been converted to assertions The outcome might be the expected behavior, or it might indicate a problem. Further review and verification is required. If you determine that the outcome reflects the expected behavior, you verify it. If not, you specify the correct outcome.

For tests run from the command line interface, tasks are reported in the **Test Generation** and **Test Execution** section of the report. If results were sent to Team Server, you can import results into the GUI as described in the Importing Results into the UI so that they appear in the Quality Tasks view.

Viewing Test Configurations that Trigger Tasks

You can view Test configurations that trigger tasks by right-clicking on a task in the Quality Tasks view and choosing **View Test Configuration**.



Quickly accessing test configuration from the from the Quality Tasks view is useful for group architects who are customizing tests and want to quickly disable settings that aren't applicable. Developers importing results from a server-based run may also need to open and review test configurations that trigger tasks.

Reviewing Stack Traces

For each unit testing problem reported, C++test reports the stack trace for the test case that caused the problem.

To review one of the lines of code referenced in the stack trace, double-click the node that shows the line number, or right-click the node and choose **Go to** from the shortcut menu. The editor will then open and highlight the designated line of code.

Reviewing Postconditions

The results of postcondition macros (all the asserted values reported) are displayed in the Quality Tasks view. These postconditions capture the state of test objects or global variables used in the test.

Any test case with reported post conditions can automatically be validated for use in regression testing. Verification 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 verification, see Verifying Test Cases for Regression Testing.

Understanding and Customizing Task Severity Categorization

Within each category, tasks are organized according to severity to help you identify and focus on the most serious issues.

If you want the Quality Tasks view to display the severity of each task, go to the pull-down menu in the Quality Tasks view, then choose **Configure Contents** and set it to show **Severity**

Reviewing Test Case Execution Details

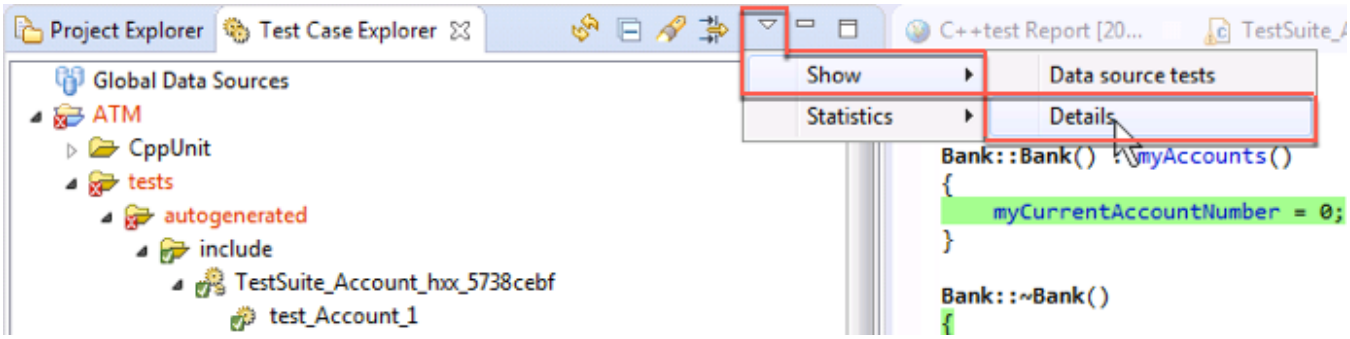
The Test Case Explorer helps you manage a project's test cases, test suites, and related data sources. It provides detailed test statistics (executed/passed /failed/skipped) and allows you to search/filter the test case tree.

By default, the Test Case Explorer is open in the left side of the UI. If it is not available, you can open it by choosing **Parasoft> Show View> Test Case Explorer**.

For details on the Test Case Explorer, see the Exploring the C++test UI.

Reviewing Test Case Details

To view test case details, enable the **Show> Details** option from the Test Case Explorer menu.

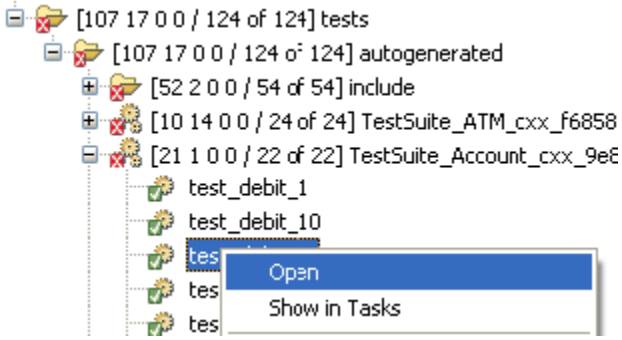


The following information will be displayed in the Test Case Explorer tree:

- Information about the function tested. This information is collected from the CPPTEST_TEST_CASE_CONTEXT comments, which C++test always adds for automatically-generated test cases and test cases created using Test Case Wizard.
- Information about the test case description. This information is collected from the CPPTEST_TEST_CASE_DESCRIPTION comments, which C++test always adds for automatically-generated test cases (see General tab) and test cases created using Test Case Wizard (see Test Case Configuration Tips).
- Information about the test execution details. Depending on test configuration (see Execution Tab Settings - Defining How Tests are Executed), this can include:
  - Test case reports / messages
  - Information about reported tasks (e.g., exceptions, failed assertions, unverified outcomes)
  - Information about checked and passed assertions

Opening the Source Code for a Test or Test Suite

To open the source code for a test suite or test case in the Test Case Explorer, right-click its Test Case Explorer node, then choose **Open**. Or, double-click its Test Case Explorer node.

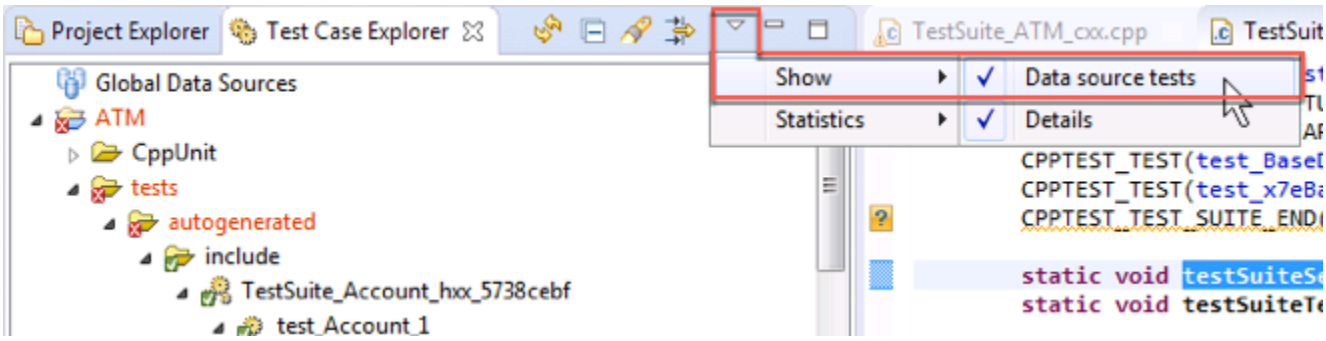


Correlating Test Case Explorer Nodes to Quality Tasks View Results

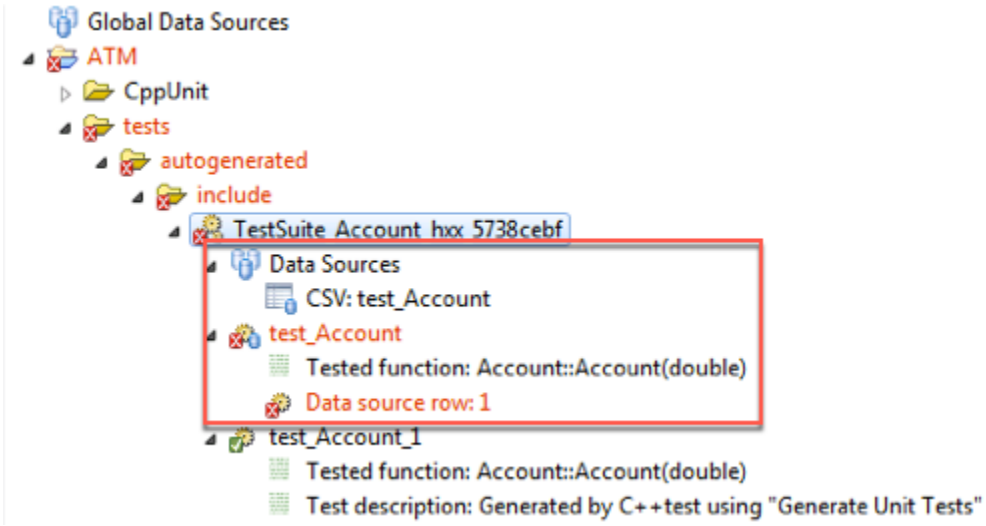
To open the test results (if available) for a test case in the Test Case Explorer, right-click its Test Case Explorer node, then choose **Show in Quality Tasks** . You can also perform the reverse action—to see the Test Case Explorer node that correlates to a result in the Quality Tasks view, right-click the Quality Tasks view node, then choose **Show in Tests**.

Reviewing Results of Data Source Test Cases

To view detailed information about executed data source test cases, enable **Show > Data source tests** from the Test Case Explorer menu. This will make C++test display information about each executed iteration of the data source test case.



The data source test case element of the tree presents statistics information about executed iterations (e.g., number of passed and failed iterations). Statistics shown for all parent nodes of the data source test case also include information about particular data source test case iterations.



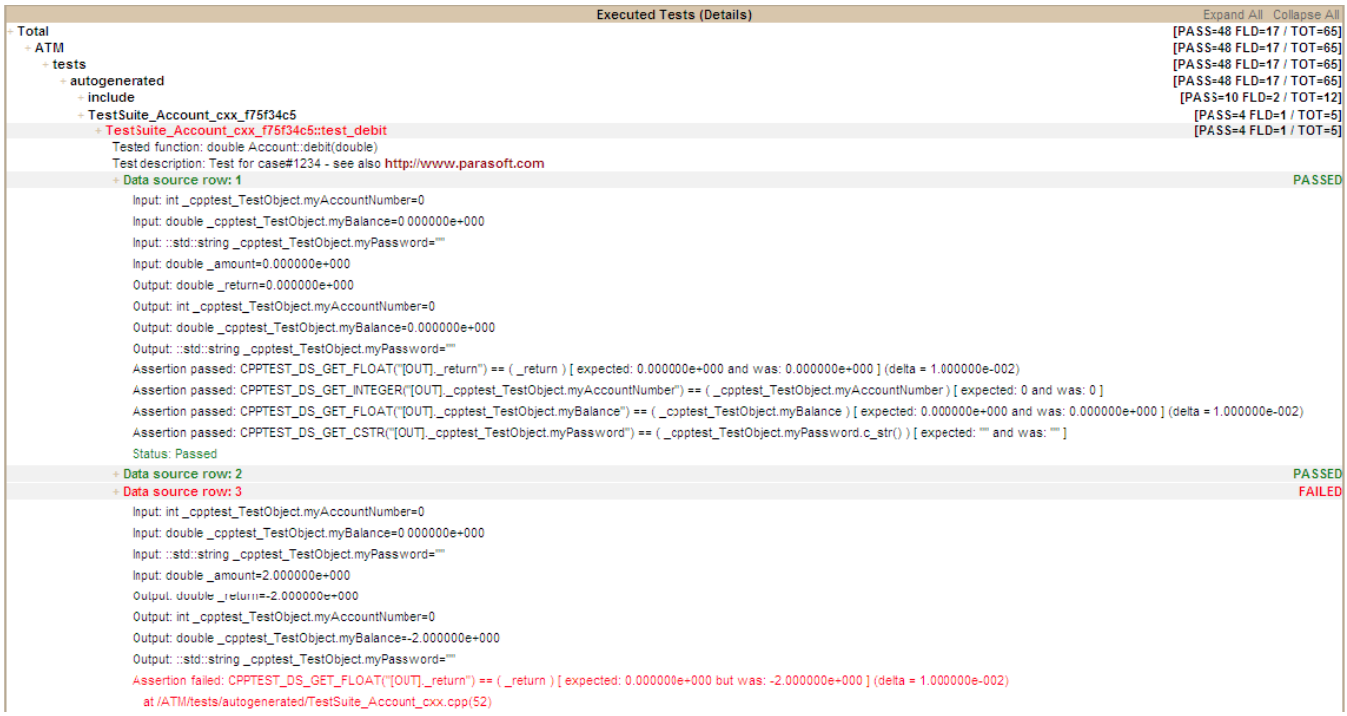
Obtaining Traceability Details

There are several ways to create a report containing the maximum amount of unit test execution and traceability information:

From here...	Perform these steps...
Generating test cases automatically	1. Provide the test case description in the **Test Configuration> Generation> General> Test case description** field. 2. Enable **Test Configuration> Generation> Test case> Insert code to report test case inputs and outputs**.
Creating test cases using Test Case Wizard	1. Provide a test case description. 2. Enable **Insert code to report test case inputs and outputs**.
Executing tests	1. Enable **Test Configuration> Execution> Runtime> Report unit test execution details**. 2. Enable **Test Configuration> Execution> Runtime> Include tasks details**. 3. Enable **Test Configuration> Execution> Runtime> Include passed assertions details**.
Generating reports	1. Enable **Parasoft> Preferences> Reports > Overview of checked files and executed tests**. 2. Choose **HTML (C++test's Unit Testing details)** as the Report format.
Generating reports from the command line	1. Set the `report.contexts_details=true` option in the localsettings file to enable the Executed Test Cases section. 2. Set the `report.format=html_ut_details` option in the localsettings file to set up the report format.

Traceability Reporting Tips

- To report additional messages from test cases, use test case report macros (see Test Case Report Macros).
- For data source test cases, reported messages might be read from the data source. For example, CPPTEST_REPORT_CSTR("Requirement number", CPPTEST_DS_GET_CSTR("req_no"))
- If a test case description or test case message contains valid URLs (e.g. http:// or ftp://), they will be added into the generated HTML report as links.



Responding to Reported Tasks

Different types of findings require different response strategies. The following table lists the categories used to classify C++test's test execution findings, and links to sections that will help you understand and respond to them.

Category	Subcategory	Description and Recommended Response
Fix Unit Test Problems	Assertion Failures	See Assertion Failures and Timeouts
	Runtime Exceptions	See Runtime Exceptions
Review Unit Test Outcomes	Unverified Outcomes	See Unverified Outcomes

Using Quick Fix (R) to Respond to Test Execution Findings

The Quick Fix (R) feature can be used to automate actions commonly performed while reviewing and responding to unit test findings.

To use Quick Fix to respond to a test execution finding:

1. Locate the detailed task message (the one with the line number) in the Quality Tasks view.
    - If you see the following icon next to the task message, it indicates that you can use Quick Fix to resolve this task:
    
2. Right-click that task message, then choose the appropriate Quick Fix option from the shortcut menu. C++test will then perform the specified action.
    - The available Quick Fix options are described below.
3. Save the modified file.

Understanding Available Quick Fix options

The following section explains the available Quick Fixes. Note that the Quick Fixes available for a specific task depends on the nature of the task.

Assertion Failure Quick Fixes

- **Verify assertion:** Indicates that the current value is correct (and the finding does not indicate a functional problem).
- **Ignore assertion:** Prevents the checking of this assertion in subsequent tests. The related test case will still be run. This action is recommended if the assertion is checking something that you don't want to check or something that changes over time (e.g., an outcome that checks the day of the month).
- **Remove Test Case:** Deletes the test case that produced given outcome—for example, if you are reviewing test results and see that a test case is invalid.

Unverified Outcome Quick Fixes

- **Verify outcome:** Indicates that the test case outcome is correct (and the finding does not indicate a functional problem).
- **Ignore outcome:** Prevents the checking of this outcome in subsequent tests. The related test case will still be run. This action is recommended if the assertion is checking something that you don't want to check or something that changes over time (e.g., an outcome that checks the day of the month).
- **Remove Test Case:** Deletes the test case that produced given outcome—for example, if you are reviewing test results and see that a test case is invalid.

Handling a Large Number of Reported Tasks (e.g., After Testing Legacy Code)

If you want to apply the same action to multiple reported problems:

1. Select the problems you want to apply the action to.
2. Right-click the selection, then choose the command for the desired response (for example, **Verify All Outcomes**).

Generating Test Execution Details Reports

The Test Execution Details report is an additional report you can generate from your regular report. It contains details about the following information:

- about tested files
- toolchain used for building test harnesses
- additional configuration files
- test suites
- test cases (including Test Case Definition and Test Case Execution Log sections)

See Generating the Test Execution Details Report for details.