Testing from the GUI

This topic explains the general procedure for running tests from the C++test GUI.

Sections include:

Prerequisites

- You must be in the C/C++ or C++test perspective in order to run tests.
- Before you can test code with C++test, it must be added to a C++test project. For instructions on creating a new project, see Creating a Project.
- Before you perform the initial test, you need to review and project options then modify them if needed. For details on how to do this, see Setting Project and File Options.
- We recommend that you prepare a customized Test Configuration before you perform tests; see Configuring Test Configurations and Rules for Policies.
- We also recommend that you configure C/C++test preferences (for task assignment, reporting, etc.) before you start testing; see C++test Configuration Overview.

If you want to exclude designated project resources from the test or test only a designated subset of project resources, you must indicate that before starting the test. See Testing a User-Defined Set of Resources for details.

Testing Headers

C++test does not directly test headers unless they are included by a source file under test.

See How do I analyze header files/what files are analyzed? for details.

Testing Template Functions

C++test does perform static analysis and unit testing of instantiated function templates and instantiated members of class templates.

See Support for Template Functions for details.

Running a Test

C++test can perform a variety of tasks, from static analysis, to unit/regression test generation and execution, to exception finding. To start using C++test to achieve your goals, you run a test based on a default or custom test scenario, which defines the precise nature and scope of C++test's analysis. These test scenarios are called "Test Configurations," and they define settings such as test scope, static analysis, test case generation, and test case execution settings.

All preconfigured Test Configurations are described in Built-in Test Configurations. The procedure for creating a custom Test Configuration is the same across the various products in the Parasoft Test family, and is described in Configuring Test Configurations and Rules for Policies.

The general procedure for testing from the GUI is as follows:

1. In the Eclipse C/C++ Projects view (a.k.a. "the project tree"), select the resource(s) that you want to test.You can use **Ctrl + click** or **Shift + click** to select multiple resources.

Running a Test Configuration

- Choose the appropriate **Test Configuration** from the **Test Using** section of the **Test** button's pull-down menu.
- Choose the appropriate **Test Configuration** from the **Parasoft> Test Using** menu in the menu bar.
- Choose the appropriate **Test Configuration** from the **Parasoft Test History** menu in the menu bar. Note that this menu contains only the most recently-run **Test Configurations**.
- Right-click the selection, then choose the appropriate **Test Configuration** from the **Test Using** shortcut menu.
- Right-click the selection, then choose the appropriate **Test Configuration** from the **Test History** shortcut menu.

"Grayed-Out" Test Configurations = Incompatible Test Configurations

If a Test Configuration is "grayed out," this indicated that it was created with an incompatible version of C++test, and cannot be applied with the current version.

For Running the #1 Favorite Test Configuration

- Click the **Test** button in the toolbar.
- Choose **Parasoft> Test Using> [Favorite Configuration]** from the menu bar.
- Right-click the resource, then choose **[product_name]> Test Using [Favorite Configuration]** from the shortcut menu.

For Running Other Favorite Test Configurations

- Click the appropriate toolbar button.
- Use the applicable keyboard shortcut. To see a list of available shortcuts, press **Shift + F9** (for Eclipse) or **Ctrl+Alt+F9** (for Visual Studio).

C++test will then run the test scenario defined by the selected Test Configuration.

Unit Testing Procedure

For unit testing, we recommend that you run multiple Test Configurations in the following order:

- Generate Unit Tests
- Generate Stubs
- Build Test Executable
- Run Unit Tests

See Generating Test Cases for Regression Testing and Exception Finding and Executing Test Cases for details.

Reviewing Results

Test progress and results summaries will be reported in the Test Progress tab that C++test opens when it starts the test. Detailed results will be reported in the Quality Tasks view, which can be opened by choosing **Parasoft> Show View> Quality Tasks**. Drill down to see details about the test findings.

The general procedure for reviewing results (as well as generating reports) is the same across the various products in the Parasoft Test family, and is described in the Reviewing Results.

Fine-Tuning Test Settings

To change test settings — such as what rules are checked, how test cases are generated, whether coverage is tracked, etc.— edit an existing Test Configuration or create a new one, then run a test using the modified/new Test Configuration. Test Configurations and all related parameters can be viewed, edited, and modified in the Test Configurations dialog. To open this dialog, choose **Parasoft> Test Configurations** from the menu bar.

For the general procedure for configuring Test Configurations, see the Configuring Test Configurations and Rules for Policies.

For details on Test Configuration settings available for C++test, see Configuring Test Configurations.

Testing a User-Defined Set of Resources

To test a user-defined set of resources in C++test:

- In the project tree, use **Ctrl+ click** or **Shift + click** to highlight the items that represent the resources you want to test, then start the test.
- Configure your preferred **Test Configuration** to limit the files and code to be analyzed via the controls available in the **Scope** tab. For details, see Scope Tab Settings: Defining What Code is Tested.

Excluding Project Resources from Testing

If you do not want all files to be analyzed/tested (for instance, to prevent the checking of automatically generated files), you can exclude project resources from testing. To indicate which project resources should not be tested:

1. In the project tree, right-click the project that contains the files that you want to be excluded.
2. Choose **Properties** from the shortcut menu, then select the **Parasoft> C++test> Scope Settings** category in the left pane.
3. Use the available controls to indicate which specific resources you want to skip and/or exclusion patterns that specify the set of resources you want to skip.

- To exclude specific files or directories, click **Add Resources**, then select the resource(s) that should not be tested.

- To specify an exclusion pattern that indicates a set of resources you want to skip, click **Add Pattern**, then enter a pattern.
    - The pattern matching is based on Ant/Nant-style wild cards: '?' matches a single character, '*' matches any string (but does not match across path segments), and '**' matches any sequence of path segments.
    - You can use the file separators / and \ interchangeably.
    - Pattern-matching is case-insensitive.
    - The paths specified are relative to the project.
        - Examples:
            - `**\Generated*.c` - Excludes all C files that 1) are in any project directory and 2) have a name starting with `Generated`.
            - `**\*.designer.c` - Excludes all C files that 1) are in any project directory and 2) have `designer.c` as the final part of the name.
            - `biz\Test*.c` - Excludes all C files that 1) are in the biz directory and 2) have a name starting with `Test`. It will not exclude such files if they are located in subdirectories of `biz`.
            - `biz\**\Test*.c` - Excludes all C files that 1) are in the `biz` directory and its subdirectories and 2) have a name starting with `Test`.

4. Click **OK** or **Apply**.

If you share your project via source control, these preferences will be shared across the team (preferences are saved in the `.parasoft` project definition file).