Performing Static Code Analysis

This topic explains how you can perform static code analysis to identify code that does not comply with a preconfigured or customized set of static analysis rules. Sections include:

Incremental Static Analysis

C/C++test is optimized to minimize the impact on your development process. By reusing information collected in previous analysis runs, it can reduce the analysis time in subsequent runs. The information is stored in the .cpptest folder in your workspace. To ensure optimal performance, avoid removing the .cpptest folder or deleting the contents of the folder.

Analyzing Headers

C++test does not directly analyze headers unless they are included by a source file under test. See How do I analyze header files/what files are analyzed? for details.

Analyzing Template Functions

C++test does perform static analysis of instantiated function templates and instantiated members of class templates. See Support for Template Functions for details.

Running Static Code Analysis

The general procedure for performing static code analysis on one or more files is as follows:

1. Select or create a Test Configuration with your preferred static code analysis settings.
   - For a description of preconfigured Test Configurations, see Built-in Test Configurations.
   - For details on how to create a custom Test Configuration, see Configuring Test Configurations and Rules for Policies. Details on C++test-specific static analysis options are available at Static Tab Settings: "Defining How Static Analysis is Performed.
2. Start the test using the preferred Test Configuration.
   - For details on testing from the GUI, see Testing from the GUI.
   - For details on testing from the command line, see Testing from the Command Line Interface.
3. Review and respond to the results.
   - For details, see Reviewing Static Code Analysis Results.
4. (Optional) Fine-tune static code analysis settings as needed.
   - For details, see Customizing Static Analysis Overview.

Configuring Batch-Mode Analysis with cpptestcli

Regularly-schedule batch-mode coding standard analysis should simply execute a built-in or custom Test Configuration that analyzes your project using the coding standard rules important to your team. For example:

- cpptestcli -data /path/to/workspace -resource "ProjectToTest" -config team://CodingStandard-sAnalysis -publish

See Testing from the Command Line Interface for more details on configuring batch-mode tests.

Detecting Duplicated Code with Static Analysis

By identifying and removing duplicate code, you make your code more concise, more readable, and easier to maintain. It can detect similar code fragments that were introduced during the development process (for example by copy-paste mistakes). It is especially useful for large projects, where manual duplication detection is tedious and ineffective.

To detect duplicated code, run the built-in "Find Duplicated Code" Test Configuration or a custom Test Configuration that includes the desired rules from the Code Duplication Detection category.

You can customize the level of code similarity that is used to determine whether two code fragments are reported as duplicates. By configuring rule properties, you can ignore variable names, string literals, number literals, and boolean literals. All text flow differences (like tabs, spaces, line breaks and comments) are always ignored.

For more details, see the rule descriptions for specific rules in the Code Duplication Detection category.