Migrating from C++test 9.x

The C++test 10.x Desktop provides full backward compatibility with C++test 9.x, and the upgrade does not require any specific migration steps. However, C++test 10.x introduces a number of enhancements and changes that may require attention. This chapter details important changes between versions 9.x and 10.x that should be taken into account when migrating from 9.x to 10.x.

Note: Parasoft offers a separate installation, **C/C++test DTP Engine**, which allows you to perform static analysis without integrating with the IDE, as well as providing the ability to execute unit tests created in open-source unit testing tools, such as GoogleTest, while monitoring code coverage for these unit tests. The Engine also provides the ability to capture code-level Application Coverage during the execution of manual tests.

In this section:

- System Requirements
- Documentation Changes
- Static Analysis
- Metrics Analysis
- Team Infrastructure
- Peer Code Reviews in the IDE vs Change Reviews in DTP
- Integration with Concerto Project Center

System Requirements

- The following operating systems are no longer supported: Solaris OS, Windows 2000, Windows XP, Windows Vista, Windows Server 2003
- C++test may require additional 1GB RAM (32bit) or 2GB RAM (64bit) for running analysis.

Documentation Changes

The C++test documentation is now available online at https://docs.parasoft.com and accessible from the **Parasoft > Help> C++** menu. (A PDF version of the online documentation is also distributed with C/C++test for offline viewing.)

Static Analysis

Reporting Code Parsing Problems

In 9.x, the "Analyze files with parse errors" option in the test configuration was disabled by default, and no results were reported for files with parse errors. In 10.x this option has been replaced with "Code Parsing Problems" with the default setting to ignore any parse problems and report static analysis findings for all files.

To retain the default behavior from C++test 9.x, set the "Code Parsing Problems" option to "Error" for a selected test configuration. See Static Tab Settings - Defining How Static Analysis is Performed for details.

Any parsing problems are reported in both HTML reports and sent to Parasoft DTP for display with the "Diagnostics> Runs - Status" widget.

Reporting Results On Cancellation

When canceling analysis prior to completion, C++test 9.x reported the partial results that were obtained. To ensure consistent results, C++test 10.x static analysis results are reported after the analysis has completed. Therefore, if the analysis is canceled before completion, no results will be reported in the IDE.

Monitoring Static Analysis Progress

The Test Progress view does not currently (as of 10.3.2) provide details about the current progress of a static analysis run. To monitor the details about a static analysis run, set the console verbosity to Normal or High and monitor the Console Output view for the current progress (in the GUI) or add the `-appconsole stdout` option to the `cpptestcli` parameters (in the command line).

Analyzing Projects with Shared Files in Eclipse

To ensure consistent results, Eclipse projects that share the same file(s) should be analyzed in separate runs (for example, if your projects include the same linked resources).

Metrics Analysis

C/C++test 10.x expands both the breadth of the code metrics calculated (see Metrics Calculation for details on the full list of available metrics) and the reporting/navigation of the metrics data – via Parasoft DTP. Calculated code metrics are consistent across all the Parasoft Engines (i.e. Jtest, dotTEST, C /C++test), and Parasoft DTP provides enhanced interfaces for aggregation and navigation of the metrics data (see the Metrics Explorer section of the DTP documentation at https://docs.parasoft.com/ for more details).

Note:

- The configuration of metrics and violation thresholds are currently (as of 10.3.2) only configurable via the Test Configuration interface in DTP.
- The IDE based metrics view and static HTML metrics reports generated by C++test are no longer available.

Team Infrastructure

Integration with Development Testing Platform

C++test 10.x can be integrated directly with the Parasoft Development Testing Platform (DTP) – the next generation of Parasoft Concerto. Parasoft DTP aggregates analysis and test results, including granular line coverage (per test case), to help you focus on the quality tasks that have the most impact. See Connecting to Development Testing Platform for information how to connect to DTP.

Connecting C++test to DTP also enables the configuration of C++test using test configurations stored, managed and edited on the DTP server; see DTP Test Configurations for details.

Note: Team Server is supported by 10.x, but is it recommend to use the DTP workflow across the team.

Reporting Results with the `-publish` Option

When executing the C/C++test Desktop in command line mode (i.e. for automated execution of Parasoft unit test or backward compatibility), results are sent to DTP with the `-publish` option. In C++test 9.x this option was used to report results to Team Server, and in C++test 10.x, publishing results to Team Server can be enabled with the `-publishteamserver` option.

Peer Code Reviews in the IDE vs Change Reviews in DTP

The Peer Code Review functionality from within the C++test 9.x IDE has been deprecated. With C++test 10.x you can review changes in the code base independently of the source control system – directly in the web interface of Parasoft DTP. The **Change Explorer** in Parasoft DTP provides additional insight into the code changes between builds by correlating the analysis data aggregated by DTP with the code changes into a single interface. See the Change Explorer section of the DTP documentation at https://docs.parasoft.com/ for more details.

Integration with Concerto Project Center

Integrations with Concerto Project Center have been depreciated in C++test 10.x. Reporting to Project Center for traceability is still available but importing of task information into the IDE is no longer available.