Built-in Test Configurations

This topic describes the preconfigured "built-in" Test Configurations that are included with C++test.

C++test includes a set of preconfigured "built-in" Test Configurations representing most common test scenarios. You can further customize these configurations as needed by copying and modifying the built-in configurations, or by creating new user-defined configurations from scratch. User-defined Test Configurations can be placed in the User-defined or Team category. User-defined Test Configurations are stored on the local machine and are available for all tests performed by the local C++test installation. Team Test Configurations are stored on the team's Team Server and can be accessed by all team members.

Static Analysis Group

This group includes universal static analysis test configurations. See Compliance Packs for test configurations that enforce coding standards

Test Configuration	Description
Recommended Rules	The default configuration of recommended rules. Covers most Severity 1 and Severity 2 rules. Includes rules in the Flow Analysis Fast configuration.
Flow Analysis Standard	Detects complex runtime errors without requiring test cases or application execution. Defects detected include using uninitialized or invalid memory, null pointer dereferencing, array and buffer overflows, division by zero, memory and resource leaks, and dead code. This requires a special Flow Analysis license option. See Introducing Built-in Flow Analysis Test Configurations for more details on Flow Analysis Test Configurations.
Flow Analysis Fast	The fast configuration uses "Shallowest" depth of analysis and runs faster than the standard and aggressive configurations. The fast configuration finds a moderate amount of problems and prevents violation number explosion. See Introducing Built-in Flow Analysis Test Configurations for more details on Flow Analysis Test Configurations.
Flow Analysis Aggressive	The aggressive option reports any suspicious code as a violation. See Introducing Built-in Flow Analysis Test Configurations for more details on Flow Analysis Test Configurations.
Effective C++	Checks rules from Scott Meyers' "Effective C++" book. These rules check the efficiency of C++ programs.
Effective STL	Checks rules from Scott Meyers' "Effective STL" book.
Modern C++ (11, 14 and 17)	Checks rules that enforce best practices for modern C++ standards (C++11, C++14, C++17).
Find Duplicated Code	Detects duplicated functions, code fragments, string literals, and #include directives.
Find Unused Code	Includes rules for identifying unused/dead code.
Metrics	Reports metrics statistics and detects metric values out of acceptable ranges.
Global Analysis	Checks the Global Static Analysis rules.
Sutter-Alexandrescu	Checks rules based on the book "C++ Coding Standards," by Herb Sutter and Andrei Alexandrescu.
The Power of Ten	Checks rules based on Gerard J. Holzmann's article "The Power of Ten - Rules for Developing Safety Critical Code." (http://spinroot.com/gerard/pdf/Power_of_Ten.pdf)

Compliance Packs

Compliance Packs include test configurations tailored for particular compliance domains to help you enforce industry-specific compliance standards and practices. See Compliance Packs Rule Mapping for information how the standards are mapped to C/C++test's rules.

Displaying compliance results on DTP

Some test configurations in this category have a corresponding "Compliance" extension on DTP, which allows you to view your security compliance status, generate compliance reports, and monitor the progress towards your security compliance goals. These test configurations require dedicated license features to be activated. Contact Parasoft Support for more details on Compliance Packs licensing.

See the "Extensions for DTP" section in the DTP documentation for the list of available extensions, requirements, and usage.

Aerospace Pack

Test Configuration	Description
Joint Strike Fighter	Checks rules that enforce the Joint Strike Fighter (JSF) program coding standards.
DO178C Software Level A Unit Testing	Executes unit tests with appropriate configuration of coverage metrics and reporting settings for DO178C Software Level A
DO178C Software Level B Unit Testing	Executes unit tests with appropriate configuration of coverage metrics and reporting settings for DO178C Software Level B
DO178C Software Level C and D Unit Testing	Executes unit tests with appropriate configuration of coverage metrics and reporting settings for DO178C Software Level C and D

Automotive Pack

Test Configuration	Description
AUTOSAR C++14 Coding Guidelines	Checks rules that enforce the AUTOSAR C++ Coding Guidelines (Adaptive Platform, version 19.03). This test configuration is part of Parasoft Compliance Pack solution that allows you to monitor compliance with industry standards using the "Compliance" extensions on DTP. It requires dedicated license features to be activated. Contact your Parasoft representative for details.
High Integrity C++	Checks rules that enforce the High Integrity C++ Coding Standard.
HIS Source Code Metrics	Checks metrics required by the Herstellerinitiative Software (HIS) group.
MISRA C 1998	Checks rules that enforce the MISRA C coding standards.
MISRA C 2004	Checks rules that enforce the MISRA C 2004 coding standards.
MISRA C++ 2008	Checks rules that enforce the MISRA C++ 2008 coding standards.
MISRA C 2012	Checks rules that enforce the MISRA C 2012 coding standards. This test configuration is part of Parasoft Compliance Pack solution that allows you to monitor compliance with industry standards using the "Compliance" extensions on DTP. It requires dedicated license features to be activated. Contact your Parasoft representative for details.
ISO26262 ASIL A Unit Testing	Executes unit tests with appropriate configuration of coverage metrics and reporting settings for ISO26262 ASIL A
ISO26262 ASIL B and C Unit Testing	Executes unit tests with appropriate configuration of coverage metrics and reporting settings for ISO26262 ASIL B and C
ISO26262 ASIL D Unit Testing	Executes unit tests with appropriate configuration of coverage metrics and reporting settings for ISO26262 ASIL D

Medical Devices Pack

Test Configuration	Description
Recommended Rules for FDA (C)	Checks rules recommended for complying with the FDA General Principles for Software Validation (test configuration for the C language).
Recommended Rules for FDA (C++)	Checks rules recommended for complying with the FDA General Principles for Software Validation (test configuration for the C++ language).

Security Pack

Test Configuration	Description

CWE Top 25 2019	Includes rules that find issues classified as Top 25 Most Dangerous Programming Errors of the CWE standard. This test configuration is part of Parasoft Compliance Pack solution that allows you to monitor compliance with industry standards using the "Compliance" extensions on DTP.
CWE Top 25 2019 + On the Cusp	Includes rules that find issues classified as Top 25 Most Dangerous Programming Errors of the CWE standard or included on the CWE Weaknesses On the Cusp list. This test configuration is part of Parasoft Compliance Pack solution that allows you to monitor compliance with industry standards using the "Compliance" extensions on DTP.
OWASP Top 10 2017	Includes rules that find issues identified in OWASP's Top 10 standard. This test configuration is part of Parasoft Compliance Pack solution that allows you to monitor compliance with industry standards using the "Compliance" extensions on DTP. It requires dedicated license features to be activated. Contact your Parasoft representative for details.
Payment Card Industry Data Security Standard	Checks rules for the security issues referenced in section 6 of the Payment Card Industry Data Security Standard (PCI DSS) (https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml) Issues detected include input validation (to prevent cross-site scripting, injection flaws, malicious file execution, etc.) and validation of proper error handling.
Security Rules	Checks rules designed to prevent or identify security vulnerabilities.
SEI CERT C Coding Guidelines	Checks rules and recommendations for the SEI CERT C Coding Standard. This standard provides guidelines for secure coding. The goal is to facilitate the development of safe, reliable, and secure systems by, for example, eliminating undefined behaviors that can lead to undefined program behaviors and exploitable vulnerabilities.
SEI CERT C Rules	Checks rules for the SEI CERT C Coding Standard. This standard provides guidelines for secure coding. The goal is to facilitate the development of safe, reliable, and secure systems by, for example, eliminating undefined behaviors that can lead to undefined program behaviors and exploitable vulnerabilities. This test configuration is part of Parasoft Compliance Pack solution that allows you to monitor compliance with industry standards using the "Compliance" extensions on DTP. It requires dedicated license features to be activated. Contact your Parasoft representative for details.
SEI CERT C++ Rules	Checks rules for the SEI CERT C++ Coding Standard. This standard provides guidelines for secure coding. The goal is to facilitate the development of safe, reliable, and secure systems by, for example, eliminating undefined behaviors that can lead to undefined program behaviors and exploitable vulnerabilities. This test configuration is part of Parasoft Compliance Pack solution that allows you to monitor compliance with industry standards using the "Compliance" extensions on DTP. It requires dedicated license features to be activated. Contact your Parasoft representative for details.
UL 2900	Includes rules that find issues identified in the UL-2900 standard.

Unit Testing Group

Test Configuration	Description
File Scope> Build Test Executable (File Scope)	Builds test executable for "trial builds." *Only the selected file(s) will be instrumented.*
File Scope> Collect Stub Information (File Scope)	Collects symbols data to populate the Stubs view. *Only the selected file(s) will be instrumented.*
File Scope> Debug Unit Tests (File Scope)	Executes unit tests under the debugger. *Only the selected file(s) will be instrumented.*
File Scope> Generate Stubs (File Scope)	Generates stubs for missing function and variable definitions. *Only the selected file(s) will be instrumented.*
File Scope> Run Unit Tests	Executes the available test cases. *Only the selected file(s) will be instrumented.*
Build Test Executable	Builds test executable for "trial builds." *All project files will be instrumented.*

Collect Stub Information	Collects symbols data to populate the Stubs view. *All project files will be instrumented.*
Debug Unit Tests	Executes unit tests under the debugger. *All project files will be instrumented.*
Generate Regression Base	Generates a baseline test suite that captures the project code's current functionality; to detect changes from this baseline, you run your evolving code base against this test suite on a regular basis. Outcomes are automatically verified.
Generate Stubs	Generates stubs for missing function and variable definitions. *All project files will be instrumented.*
Generate Test Suites	Generates test suites (without generating test cases) for the selected resources.
Generate Unit Tests	Generates unit tests for the selected resources.
Run Unit Tests	Executes the available test cases. *All project files will be instrumented.*
Run Unit Tests with Memory Monitoring	Executes the available test cases and collects information about memory problems. *All project files will be instrumented.*

Application Monitoring Group

Test Configuration	Description
Build Application with Coverage Monitoring	Builds the tested application with coverage monitoring enabled.
Build Application with Full Monitoring	Builds the tested application with coverage and memory monitoring enabled.
Build Application with Memory Monitoring	Builds the tested application with memory monitoring enabled.
Build and Run Application with Coverage Monitoring	Builds and executes the tested application with coverage monitoring enabled.
Build and Run Application with Full Monitoring	Builds and executes the tested application with coverage and memory monitoring enabled.
Build and Run Application with Memory Monitoring	Builds and executes the tested application with memory monitoring enabled.

Utilities Group

Test Configuration	Description
Load Test Results (File)	Used to collect test results via the file channel. By default, this configuration assumes that logs are located inside `${cpptest:testware_loc}`. If needed, you can customize this location to any file system location that can be accessed from the C++test GUI.
Load Test Results (Sockets)	Used for "on the fly" collection of test results sent through TCP/IP sockets. It starts a java utility program to listen to and capture test results. You can customize the port numbers for test and coverage results. Port numbers are defined with the `results_port` and `coverage_port` properties.
Extract Library Symbols	Used to extract a list of symbols from external libraries (or object files). It should be used whenever C++test's standard algorithm for collecting information about symbols from binaries is not sufficient. For example if you use a Wind River DKM type of project, you may want to have all symbols from the VxWorks image collected in this way. You will probably need to enter the location of the binaries you want to extract symbols from, as well as the name of the nm-like utility that can be used to dump the content of library /object file.
Generate Stubs Using External Library Symbols	Used to generate stubs after the "Extract Library Symbols" Test Configuration has been run. It assumes that a file with a list of symbols from external libraries is stored in the project temporary data.
Load Application Coverage	Used to import the coverage data collected with the `cpptestcc` coverage tool into your IDE; see Collecting Application Coverage with cpptestcc.

Load Archived Results	Used to load the archived results into C/C++test; see Merging Results from Multiple Test Runs.

See Configuring Test Configurations and Rules for Policies to learn how to develop custom Test Configurations that are tailored to your projects and team priorities.

Compliance Packs Rule Mapping

This section includes rule mapping for the CWE standard. The mapping information for other standards is available in the PDF rule mapping files shipped with Compliance Packs.

CWE Top 25 Mapping

CWE ID	CWE Name	Parasoft rule ID(s)
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	• CWE-119-a • CWE-119-b • CWE-119-c • CWE-119-d • CWE-119-e • CWE-119-f • CWE-119-g • CWE-119-h • CWE-119-i • CWE-119-j
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	N/A
CWE-20	Improper Input Validation	• CWE-20-a • CWE-20-b • CWE-20-c • CWE-20-d • CWE-20-e • CWE-20-f • CWE-20-g • CWE-20-h • CWE-20-i • CWE-20-j
CWE-200	Information Exposure	• CWE-200-a
CWE-125	Out-of-bounds Read	• CWE-125-a • CWE-125-b • CWE-125-c • CWE-125-d
CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	• CWE-89-a
CWE-416	Use After Free	• CWE-416-a • CWE-416-b • CWE-416-c
CWE-190	Integer Overflow or Wraparound	• CWE-190-a • CWE-190-b • CWE-190-c • CWE-190-d • CWE-190-e • CWE-190-f • CWE-190-g

CWE-352	Cross-Site Request Forgery (CSRF)	N/A
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	• CWE-22-a
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	• CWE-78-a
CWE-787	Out-of-bounds Write	• CWE-787-a • CWE-787-b • CWE-787-c • CWE-787-d • CWE-787-e • CWE-787-f
CWE-287	Improper Authentication	• CWE-287-a
CWE-476	NULL Pointer Dereference	• CWE-476-a • CWE-476-b
CWE-732	Incorrect Permission Assignment for Critical Resource	• CWE-732-a • CWE-732-b
CWE-434	Unrestricted Upload of File with Dangerous Type	N/A
CWE-611	Improper Restriction of XML External Entity Reference	• CWE-611-a
CWE-94	Improper Control of Generation of Code ('Code Injection')	N/A
CWE-798	Use of Hard-coded Credentials	• CWE-798-a
CWE-400	Uncontrolled Resource Consumption	• CWE-400-a
CWE-772	Missing Release of Resource after Effective Lifetime	• CWE-772-a • CWE-772-b
CWE-426	Untrusted Search Path	• CWE-426-a
CWE-502	Deserialization of Untrusted Data	N/A
CWE-269	Improper Privilege Management	• CWE-269-a • CWE-269-b
CWE-295	Improper Certificate Validation	N/A

CWE Weaknesses On the Cusp Mapping

CWE ID	CWE Name	Parasoft rule ID(s)
CWE-835	Loop with Unreachable Exit Condition ('Infinite Loop')	• CWE-835-a
CWE-522	Insufficiently Protected Credentials	N/A

CWE-704	Incorrect Type Conversion or Cast	• CWE-704-a • CWE-704-b • CWE-704-c • CWE-704-d • CWE-704-e • CWE-704-f • CWE-704-g • CWE-704-h • CWE-704-i • CWE-704-j • CWE-704-k • CWE-704-l
CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	• CWE-362-a • CWE-362-b • CWE-362-c • CWE-362-d • CWE-362-e
CWE-918	Server-Side Request Forgery (SSRF)	N/A
CWE-415	Double Free	• CWE-415-a
CWE-601	URL Redirection to Untrusted Site ('Open Redirect')	N/A
CWE-863	Incorrect Authorization	• CWE-863-a
CWE-862	Missing Authorization	N/A
CWE-532	Inclusion of Sensitive Information in Log Files	• CWE-532-a
CWE-306	Missing Authentication for Critical Function	N/A
CWE-384	Session Fixation	N/A
CWE-326	Inadequate Encryption Strength	• CWE-326-a
CWE-770	Allocation of Resources Without Limits or Throttling	• CWE-770-a
CWE-617	Reachable Assertion	• CWE-617-a