

Configuring Test Configurations

In this section:

- [Overview](#)
- [Running a Test Configuration](#)
- [Viewing Available Test Configurations](#)
- [Built-in Test Configurations](#)
- [Creating Custom Rules](#)

Overview

Test configurations define how your code is analyzed and tested, including which static analysis rules are enabled, which tests to run, and other analysis parameters. C/C++ test ships with built-in test configurations, but users can create and store their own test configurations in the DTP server (see the DTP documentation for details).

User-defined test configurations that are stored in DTP can be downloaded from the DTP server and stored in the [INSTALL_DIR]/configs/user directory as *.properties files.

Running a Test Configuration

You can specify which configuration will be run in one of the following ways:

- Run `cpptestcli` with the `-config` switch and specify a built-in, user-defined or DTP-hosted test configuration:

```
-config "builtin://Recommended Rules"  
-config "user://Foo Configuration"  
-config "dtp://Foo Team Configuration"  
-config "dtp://FooTeamConfig.properties"
```

You can also provide a path or URL to the test configuration .properties file:

```
-config "C:\Devel\Configs\FooConfig.properties"  
-config "http://foo.bar.com/configs/FooConfig.properties"
```

For example, your command line may resemble the following:

```
cpptestcli -config "builtin://Recommended Rules" -compiler gcc_3_4 -input cpptest.bdf
```

- In the .properties file, specify the default configuration that will be run when the `-config` option is not used:

```
cpptest.configuration=user://Configuration Name
```

Viewing Available Test Configurations

Use the `-listconfigs` switch to print the available test configurations.

Built-in Test Configurations

The following tables include the test configurations shipped in the [INSTALL]/configs/builtin directory.

Static Analysis

This group includes universal static analysis test configurations. See [Compliance Packs](#) for test configurations that enforce coding standards.

| Built-in Test Configuration | Description |
|-----------------------------|---|
| 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. |
| Find Duplicated Code | Applies static code analysis rules that report duplicate code. Duplicate code may indicate poor application design and lead to maintainability issues. |
| Find Unused Code | Includes rules for identifying unused/dead code. |
| 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. |
| Flow Analysis Aggressive | Includes rules for deep flow analysis of code. A significant amount of time may be required to run this configuration. |
| Flow Analysis Fast | Includes rules for shallow depth of flow analysis, which limits the number of potentially acceptable defects from being reported. |
| Global Analysis | Checks the Global Static Analysis rules. |
| Metrics | Computes values for several code metrics. |
| Modern C++ (11, 14 and 17) | Checks rules that enforce best practices for modern C++ standards (C++11, C++14, C++17). |
| Recommended Rules | The default configuration of recommended rules. Covers most Severity 1 and Severity 2 rules. Includes rules in the Flow Analysis Fast configuration. |
| 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.

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

| Built-in Test Configuration | Description |
|-----------------------------|--|
| Joint Strike Fighter | Checks rules that enforce the Joint Strike Fighter (JSF) program coding standards. |

Automotive Pack



| Built-in Test Configuration | Description |
|---------------------------------|---|
| AUTOSAR C++14 Coding Guidelines | Checks rules that enforce the AUTOSAR C++ Coding Guidelines (Adaptive Platform, version 17-10).  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. |
| HIS Source Code Metrics | Checks metrics required by the Herstellerinitiative Software (HIS) group. |

| | |
|--------------------|---|
| High Integrity C++ | Checks rules that enforce the High Integrity C++ Coding Standard. |
| MISRA C 1998 | Checks rules that enforce the MISRA C coding standards |
| MISRA C 2004 | Checks rules that enforce the MISRA C 2004 coding standard. |
| MISRA C 2012 | Checks rules that enforce the MISRA C 2012 coding 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. |
| MISRA C++ 2008 | Checks rules that enforce the MISRA C++ 2008 coding standards. |

Medical Devices Pack

| Built-in 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

| Built-in Test Configuration | Description |
|---|--|
| CWE-SANS Top 25 Most Dangerous Programming Errors | Includes rules that find issues classified as Top 25 Most Dangerous Programming Errors of the CWE-SANS standard. |
| OWASP Top 10 2017 | Includes rules that find issues identified in OWASP's Top 10 standard |
| Payment Card Industry Data Security Standard | Includes rules that find issues identified in PCI Data Security Standard |
| SEI CERT C 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. |
| Security Rules | General test configuration that finds security issues |
| UL 2900 | Includes rules that find issues identified in the UL-2900 standard. |

Runtime Analysis

| Built-in Test Configuration | Description |
|-----------------------------|-------------------------------------|
| Coverage | Generates the code coverage report. |

| | |
|--------------|---|
| GoogleTest | Analyzes Google Test unit test results. |
| Unit Testing | Analyzes CppUnit or CppUTest test results collected with C/C++test's connector (see Integrating with CppUnit and CppUtest) |

Creating Custom Rules

Use RuleWizard to create custom rules. To use the rule, it needs to be enabled in a test configuration and the custom rule file must be located in the [INSTALL_DIR]\rules\user\ directory, or another user-specific directory.