Built-in Static Analysis Rules

This topic describes the preconfigured "built-in" static analysis rules that are included with C++test.

Sections include:

Understanding Rule Categories

C++test provides hundreds of code that industry experts have developed to help identify potential bugs from improper C/C++ language usage, enforce best coding practices, and improve code maintainability and reusability. These rules are organized into the following categories:

- BugDetective [BD]
- Coding Convention [CODSTA]
- Code Duplication Detection [CDD]
- Comments [COMMENT]
- Exception [EXCEPT]
- Formatting [FORMAT]
- Initialization [INIT]
- Joint Strike Fighter [JSF]
- Metrics [METRICS]
- Misra C[MISRA]
- Misra C 2004 [MISRA2004]
- Misra C++ 2008 [MISRA2008]
- Memory and Resource Management [MRM]
- Naming Conventions [NAMING]
- Object Oriented [OOP]
- Optimization [OPT]
- Possible Bugs [PB]
- Physical File Organization [PFO]
- Portability [PORT]
- Preprocessor [PREPROC]
- Qt Best Practices [QT]
- Security [SECURITY]
- STL Best Practices [STL]
- Template [TEMPL]

Viewing Rule Descriptions

To view descriptions of all the static analysis rules that are included with C++test, choose **Parasoft> Help**, open the **C++test Static Analysis Rules** book, then browse the available rule description files

To view a list of all static analysis rules that a given Test Configuration is configured to check, as well as descriptions of these rules:

1. Open the Test Configurations dialog by choosing **Parasoft> Test Configurations** or by choosing **Test Configurations** in the drop-down menu on the **Test Using** toolbar button.
2. Select the Test Configuration that you want a rule list for.
3. Open the **Static** tab.
4. Click **Printable Docs**.

If you want to print the list of rules and all related rule descriptions, enable your browser's **Print all linked documents** printer option before you print the main the list of rules.

Severity Levels

Each rule is assigned a severity level. The severity level indicates the chance that a violation of the rule will cause a serious construction defect (a coding construct that causes application problems such as slow performance, memory leaks, security vulnerabilities, and so on). Possible severity levels (listed from most severe to least severe) are:

- Highest - Level 1
- High - Level 2
- Medium - Level 3
- Low - Level 4
- Lowest - Level 5

Custom Rules

C++test can also check any number of custom rules that you design with the RuleWizard module. With RuleWizard, rules are created graphically (by creating a flow-chart-like representation of the rule) or automatically (by providing code that demonstrates a sample rule violation). By creating and checking custom rules, teams can verify unique project and organizational requirements, as well as prevent their most common errors from recurring.

RuleWizard can be used to modify built-in static code analysis and to add additional ones. For more information about creating custom coding rules, see Customizing Existing Rules and Creating New Rules.

Preference Settings

This topic describes C++test-specific settings that can be configured in the Parasoft Preferences panel.

Sections include:

- General Preference - Ignoring Project Containers in Paths
- File Encoding Settings

How to Configure Preferences and Share all Preferences Across the Team

The general procedures related to configuring and sharing preferences are standardized across a number of Parasoft products. Moreover, many configuration details (source control, task assignment, Team Server connection, reporting, etc.) are also standardized, and can be centrally configured in the top-level Parasoft branch of the Preferences tree. For details, see C++test Configuration Overview.

General Preference - Ignoring Project Containers in Paths

If you select the top-level node you can specify if you want to **Ignore solution and solution folder names in paths**. *If you change this setting, we strongly recommend that you restart the IDE to properly restructure the solutions and projects.*

Enabling the **Ignore solution and solution folder names in paths** option is recommended for teams where:

- The main development unit is a project.
- The developers don't have any conventions for solutions and don't share solution settings.
- Many solutions contain the same project.
- Developers want to share project and resource settings used in many different solutions.

Paths in this mode:

```
/User Interface/Program.cs
/Connector/Program.cs
```

We recommend leaving this option disabled when:

- The main development unit is a solution.
- The solutions are shared by developers.
- Each solution should have separately-controlled settings for projects and resources.

Paths in this mode:

```
/My Solution/Common/User Interface/Program.c
/My Solution/Engine/Connector/Program.c
```

This setting will be shared across the team by Team Server (if available). If the team has just isolated desktop installations (not connected to Team Server), this option can be locally controlled by each user and can be set in all editions.

File Encoding Settings

The File Encoding panel allows you to determine the encoding for files without a BOM signature. You can choose to use the system defaults, indicate specific encoding, or automatically detect a specific Far-East language.