Customizing Existing Rules and Creating New Rules

This topic explains how to check custom requirements or tailor existing rules to your unique needs by either modifying existing rules or by creating custom rules.
Sections include:

Introduction

A cornerstone of static analysis with C++test is the ability to customize both static analysis Test Configurations (the set of rules to check) and specific rules—including creating new custom rules.

The rules provided with C++test are a comprehensive set to select from, yet specific coding guidelines may differ slightly, or require well-defined exceptions to the rules, depending on the style of coding, or the nature of the application. Hence, it is very important to understand how C++test rules can be customized and deployed to a team of users.

C++test has the following types of rules:

- **Pattern-based rules** check code for specific patterns that are encoded in the rule, and trigger when the code matches the pattern. This type of analysis is usually local to a compilation unit (a single source file with all included headers). Many pattern-based rules can be parameterized using RuleWizard (see Customizing Rules with RuleWizard).
- **Flow-based (Flow Analysis) rules** check coding patterns and flow of data across multiple functions and source files. Many flow-based rules can be parameterized by changing their configuration parameters directly in the C++test GUI (see Customizing Parameterized Rules). RuleWizard is not used to configure flow-based rules.

Customizing Parameterized Rules

A number of rules are parameterized, meaning that you can customize the nature of the rules by modifying the available rule parameters. Parameterized rules are marked with a special icon (a wizard hat with a radio button) in the Test Configurations dialog **Static> Rules Tree** tab:

Available rule parameters are described in the rule's documentation. To view a rule's description, right-click the node that represents that rule, then choose **View Rule Documentation** from the shortcut menu.

To modify rule parameters:

1. Open the Test Configurations dialog by choosing **Parasoft> Test Configurations** or by choosing **Test Configurations** in the drop-down menu on the **Run Tests** toolbar button.
2. Open the **Static> Rules Tree** tab for any Test Configuration. The modified rule parameters will be applied to all Test Configurations, so it does not matter which Test Configuration you select in this step.
3. Expand the rule's category branch.
4. Right-click the parameterized rule that you want to modify, then choose **View/Change Rule Parameters** from the shortcut menu.
5. Modify the rule parameters in the dialog that opens.
6. Click **OK** to save your changes.

Creating Multiple Versions of a Parameterized Rule

If you want to create multiple versions of a parameterized rule (such as a Flow Analysis rule that looks for user-specified definitions of leaks), create a "clone" of that rule (as described in Specifying Rule Mappings) then parameterize each of the rule instances.

Customizing and Creating Rules with RuleWizard

This section explains the general workflow for customizing existing rules and creating new rules with RuleWizard. See RuleWizard User Guide for details about RuleWizard capabilities and usage.

Customizing Rules with RuleWizard

Many of C++test's pattern-based rules are edited with RuleWizard. C++test rules are physical files that can be loaded in RuleWizard and changed as needed. The built-in C++test rules are included in the installation directories.

Customizable rules are marked with the following wizard hat + wizard wand icon in the Test Configuration panel's rules tree: 

We strongly recommend that you leave the C++test built-in rules intact. To customize the rule, duplicate a built-in rule, and then modify the duplicate.

To customize a rule with RuleWizard:

1. Open the Test Configurations dialog by choosing **Parasoft> Test Configurations** or by choosing **Test Configurations** in the drop-down menu on the **Run Tests** toolbar button.
2. Open the **Static> Rules Tree** tab for any Test Configuration.
3. Right-click the rule you want to modify, then choose **Duplicate** from the shortcut menu.
4. A duplicate rule node—with a file icon—will be added to the rules tree.
    - A copy of the rule file(s) will be added to the user-specific disk location. The rule ID changes with this operation so that the duplicated rule does not mask the built-in rule (see Note on Duplicated Rule IDs for details)
5. Right-click the duplicate rule, then choose **Edit Rule in RuleWizard** from the shortcut menu to open the rule in RuleWizard.
6. Edit the rule in RuleWizard, and save it. See the RuleWizard User Guide (accessible by choosing **Help> Documentation** in the RuleWizard GUI) for information on how to modify and save custom rules
7. Right-click the edited rule and choose **Upload to Team Server**. This makes it accessible to other team members.
8. Enable the edited rule in the appropriate configurations.

All of the above steps can be controlled by menu options in C++test; you do not need to perform any actual file manipulations. C++test completely manages all rule components in these steps.

Creating New Rules with RuleWizard

You can easily create your own rules using the RuleWizard module–a graphical rule creation and customization tool available in C++test.

With RuleWizard, rules can be created graphically (by creating a flow-chart-like representation of the rule) or automatically (by providing code that demonstrates a sample rule violation).

To open RuleWizard:

- Choose **Parasoft> Launch RuleWizard.**

The RuleWizard GUI will then open. See the RuleWizard User Guide (accessible by choosing **Help> Documentation** in the RuleWizard GUI) for information on how to modify and save custom rules.

Deploying Customized Rules or Fully-Custom Rules

Before you can check custom rules, you must deploy them to C++test. The recommended deployment of customized or fully-custom rules leverages Team Server. As a rule, Parasoft infrastructure deployment relies on Team Server as a key component. C++test automatically connects to the Team Server using communication through a designated port. To set up your Team Server connection, go to **Parasoft> Preferences> Team Server**. See the Connecting to Team Server for additional details.

It is generally possible to deploy custom rules without Team Server, but it is notably more difficult to manage that deployment. If teams need to deploy custom rules, we strongly encourage the use of Team Server.

For details on how to use Team Server to deploy rules across the team, see Configuring Test Configurations and Rules for Policies.

Notes on Rule IDs

Each rule that you import into the tool must have a unique rule ID. You should not import multiple rules that have the same rule ID. See Note on Duplicated Rule IDs for details.

To specify rule categories using the rule ID, use the format `<rule_category>-<rule_uid>`. For example, "myrule-123" will be automatically assigned to "myrule" category.

Deploying Custom Rules (For Teams Using Team Server)

See Configuring Test Configurations and Rules for Policies.

Deploying Custom Rules (For Teams Not Using Team Server)

Before you can check custom coding rules that were designed in RuleWizard, you need to deploy them.

Note

The following procedure describes how to enable custom rules if you are not using Parasoft Team Server to share rules across the team. If you are using Team Server, follow the instructions in Configuring Test Configurations and Rules for Policies.

To deploy custom rules if you are not using Team Server:

1. Open the Test Configurations dialog by choosing **Parasoft> Test Configurations** or by choosing **Parasoft> Test Configurations** in the drop-down menu on the **Run Tests** toolbar button.
2. Select any Test Configurations category. The new rule(s) will be available in all available Test Configurations.
3. Open the **Static> Rules Tree** tab.
4. If any new rules should belong to a new category, create a new category as follows:
    1. Click the **Edit Rulemap.** button.
    2. Open the **Categories** tab.
    3. Click **New**. A new entry will be added to the category table.
    4. Enter a category ID and category description in the new entry. For instance, an organization might choose to use ACME as the category ID and ACME INTERNAL RULES as the description.
    5. Click **OK** to save the new category
5. Click the **Import** button to the right of the rules tree. The Import RuleWizard rule dialog will open.
6. Use the Import RuleWizard rule dialog to specify which rule(s) you want to import, and whether you want to overwrite existing rule files (if an imported rule file has the same name as an existing rule file).
7. Click **OK**. The rule will be displayed under the assigned category and will be disabled by default.
8. Enable the new rule(s) you want checked.
9. Click either **Apply** or **Close** to commit the modified settings.

Deploying Custom Rules on DTP

See the Managing Custom Rules section in the DTP User Guide for information how to deploy custom rules on DTP.

Note on Duplicated Rule IDs

Rules are recognized by rule ID within C++test If you have a rule with the same ID as a rule already in the built-in, user, or team categories, then the priority of usage is team/user/built-in. As a result, if you import a rule that matches the ID of a built-in rule, then the imported rule will be physically located in the default rule directory set in the Preferences panel, and the rule configuration will be use the imported rule instead of the built-in rule with the same name. If a rule is uploaded to Team Server, then the rule on Team Server will mask any user rules with the same ID, as well as the built-in rule with the same ID.