Suppressing the Reporting of Acceptable Violations

This topic explains how to prevent C++test from reporting specific static analysis violations (e.g., when you generally follow a rule, but decide to ignore that rule in an isolated number of exceptional situations). Suppression schemes can be entered in the GUI or defined directly in the source code.

Sections include:

- About Suppressions
- Defining Suppressions in the GUI
- Defining Suppressions in Source Code
    - Additional Notes
- Viewing GUI-Based Suppressions
    - Using the Suppression Filter to Restrict the Suppressions Shown
- Clearing GUI-Based Suppressions

About Suppressions

See Suppressions.

Defining Suppressions in the GUI

To suppress a static analysis task that is shown in the Quality Tasks view:

1. Right-click the Quality Tasks view item that represents the task you want to suppress, then choose **Suppress Task** from the shortcut menu.
    - To suppress all tasks in a group (a rule category, a specific rule, a file, etc.) right-click the node that represents that group, then choose **Suppress All Tasks.**
2. Enter the reason for the suppression in the dialog box that opens
3. If you want the suppression stored in the source code (as described in the following section), choose **Suppress in Source Code**. Otherwise, the suppression will be saved in Team Server (when available) or with the local installation
    - Note that if you are using the **Suppress in Source Code** option, the comment added will suppress all violations of the violated rule. In other words, if one line of code has two violations of the same rule, both of these violations—as well as any subsequent violations of this same rule—will be suppressed.

The task will then be "suppressed" and removed from the Quality Tasks view. A suppression entry will be added to the Suppressions view or directly to the source code (depending on selected mode). If the same static analysis violation is found in subsequent tests of this project, it will be reported in the Suppressions view, but not in the Quality Tasks view.

Tip

Team suppressions (as opposed to in-code suppressions) are message-based and not rule-based. Suppressions prevent the reporting of a specific static analysis task (e.g., fix the violation of rule X that occurs in line Y); they do not prevent the reporting of all violations of a rule.

Defining Suppressions in Source Code

When suppressions are defined in source code:

- You ensure that the same suppressions are applied whenever you or a team member tests that code.
- You can add code comments explaining each suppressions, so the reason for each suppression is always clear when you or team members are reviewing the code.
- You gain fine-grained control over which rules are enforced at the file, class, or line level.

There are two ways to define suppressions in source code:

- Enter them from the GUI, then choose **Suppress in Source Code**—this approach is described above.
- Enter them directly in the source code—this approach is described below.

C++test allows you to define suppressions directly in the source code, using the following directives embedded in C or C++ style comments:

1. `parasoft-suppress <ruleid> [<ruleid> ...] ["<comment>"]:` Creates a suppression for the current line only (line with suppression directive)
2. `parasoft off`: Suppresses the reporting of all static analysis violations that occur between this line of code and the end of the file.
3. `parasoft on:` Unsuppresses the reporting of all static analysis violations that occur between this line of code and the end of the file.
4. `parasoft suppress/unsuppress [line <linewildcard>][class <classnamewildcard>][file <filenamewildcard>] [item <ruleid>][type <severity>] [reason <comment>]:`
   Creates a suppression with the desired parameters.
    - Examples:
        - `line *, line 10, line 50-*`

    - `class MyClass, class *`
    - `file *example.cpp, file *ex*.cpp`
    - `item SECURITY-02`
    - `type SV`
    - `reason "Not relevant to my code."`
  - The above elements (`line, class, file`, etc.) can be combined together by separating each entry with a space (for example, `item * class MyClass line 10-*`).
  - `<filenamewildcard>` must be specified as an absolute path to a file.

Example A:
`/* parasoft suppress item SECURITY-02 */`
This suppresses reporting of violation SECURITY-02 for the current file, starting from the line where the suppression is located.

Example B:
`/* parasoft unsuppress item * class MyClass line 10-* */`
This unsuppresses all items for class `MyClass` in the current file starting from line 10.

Example C:
`/* parasoft unsuppress item SECURITY-02 file *example.cpp */`
This unsuppresses the item SECURITY-02 in any file that is in the current compilation unit and has a file path that matches the *example.cpp file name pattern.

Example D:
`/* parasoft suppress all */`
This suppresses all messages in the current file starting from the line where this directive was entered.

Example E:
```
void foo(); /* parasoft-suppress SECURITY-02 */
void bar(); // parasoft-suppress SECURITY-03 "Not relevant to my code."
```
This suppresses item SECURITY-02 on the line with "foo" declaration and item SECURITY-03 on the line with "bar" declaration with the reason "Not relevant to my code."

Example F:
```
/* parasoft off */
/* parasoft unsuppress line 30-45 */
```
This suppresses everything, then unsuppress lines 30 to 45.

Additional Notes

- When no `item` is specified, the (un)suppression will be applied to all rules. This is equivalent to specifying `item *`.
- When no `file` is specified, the (un)suppression will be applied only to the current file.
- When no `line` is specified, the (un)suppression will be applied starting from the line that contains the directive.
- To suppress files with whitespaces in their names, quotes should be used. For example: `parasoft suppress file "*file with whitespace.cpp"`
  Putting a suppression with the `file` * pattern in a header file makes it valid for all the files that include that header file. When no `file` is specified, suppression will be applied only to that header.
- The `parasoft-suppress` directive is quite distinct from the other `parasoft on, parasoft off`, and `parasoft suppress/unsuppress` directives. `parasoft-suppress` suppresses only the current line (the line with the directive) and uses different syntax than the other directives.

Viewing GUI-Based Suppressions

To view the suppressed messages that were reported in a subsequent test run:

1. Open the Suppressions view. If this view is not available, choose **Parasoft> Show View> Suppressions** to open it.
2. Select the resource whose suppressions you want to view.

The Suppression Messages view will display the following information:

- The static analysis violation that was suppressed.
- The reason why the task was suppressed.
- The resource (file) to which the suppression applies.
- The folder that contains the resource.
- The name of the person who suppressed the task.
- The date on which the suppression was first applied.

To sort the Suppressions view contents by one of the column headings, click that column heading.

Tip

You can edit a suppression's message or reason by right-clicking the suppression in the Suppressions view, choosing **Edit Message** or **Edit Reason** from the shortcut menu that opens, then modifying the message or reason in the dialog that opens.

Using the Suppression Filter to Restrict the Suppressions Shown

You can restrict which suppressions are shown in the Suppressions view by using the available suppressions filter.

To filter the suppressions shown in the Suppressions view:

1. Click the **Filter** button in the Suppressions view toolbar. The Filters dialog will open.
2. Check the **Enabled** check box to enable the filter.
3. Use the dialog's controls to specify your filtering criteria. Available options include:
    - **Limit visible items to**: Shows no more than the specified number of suppressions.
    - **On any resources:** Shows all suppressions for all projects.
    - **On any resource in same project:** Shows all suppressions for the currently-selected project.
    - **On selected resource only**: Shows only the suppressions entered for the currently-selected resource.
    - **On selected resource only and its children:** Shows only the suppressions entered for the currently-selected resource, and that resource's children.

Clearing GUI-Based Suppressions

To unsuppress a task:

- Select the Suppressions view item that represents the task you want to unsuppress, then click the Red **X Remove Suppression** icon in the top right of the view.

If this same static analysis violation is found in subsequent tests of this project, the task will be reported in the Quality Tasks view.