Configuring and Running Pre-Commit Code Review Scans

This topic explains how to set up and run a pre-commit code review process where developers submit code for review *before* it is committed to source control. Each time a developer needs their code reviewed, he or she runs the Code Review Test Configuration. This Test Configuration detects code changes and prepares them for review.

Section include:

Related Topics

- For details on configuring **post-commit code review**, which is for teams who want to review code after it is committed to source control, see Configuring and Running Post-Commit Code Review Scans.
- For details on pre-commit vs. post-commit vs. task-driven workflows, see Workflow Overview.
- For details on general code review configuration options (options that apply to both pre-commit and post-commit processes, see General Code Review Configuration.

Configuration Overview

Configuring a pre-commit code review requires:

- Setting up the appropriate preferences on all team installations as described in General Code Review Configuration.
- Configuring a Code Review Test Configuration on one machine and sharing it across the team as described below

Configuring a Test Configuration for Pre-Commit Code Reviews

The Code Review Test Configuration detects code changes and prepares them for review. We recommend configuring a single Test Configuration and sharing it across the team using Team Server. This streamlines configuration and updating.

You can create a Test Configuration that is dedicated to Code Review, or have a Test Configuration that checks for Code Reviews as well as performs other analyses.

To configure a Test Configuration that detects code changes and prepares them for review:

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. Duplicate an existing Test Configuration (such as Built-in> Code Review> Pre-Commit) or create a new one.
3. In the **Scope** tab, choose **Test only files added or modified locally**.
4. Also in the **Scope** tab, review the **File Filters> Path** options settings and adjust them if desired. For details on these Test Configuration settings, see Configuring Test Configurations.
5. If your reporting preferences are not already set to use a unique session tag for code review scans, go to the **Common** tab, enable **Override Session Tag**, then choose one of the preconfigured identifiers, or specify your own. This is the tag that will be assigned to all code reviews that stem from this Test Configuration.
    - *To ensure proper code review reporting, you must configure a session tag for your code review scans and use that session tag in DTP to specify which code reviews are associated with particular DTP projects.*
6. At the top of the **Code Review** tab:
    1. Check **Enable Code Review Scanner**.
    2. If you want the report to include code review results from all available team scanners, enable **Generate comprehensive report (includes all scanners)**. This option is not typically used for pre-commit code reviews. If this is not enabled, the report will include only results for the specified session tag.
    3. Enable **Auto publish reviews** if you want review tasks to be "published" (uploaded) automatically after this Test Configuration is run. This is recommended for pre-commit code reviews.
    4. From the **Generate tasks with priority** box, indicate the priority that should be assigned to all code review tasks that are created using this Test Configuration.

7. (Optional) If you want the reviews automatically assigned, use the **Authors**, **Reviewers**, **Monitors**, and **Filters** tabs to define how you want your code reviews assigned. Reviewers and monitors can be assigned to specific authors, or to specific project areas. If you do not specify options here, each author needs to select the appropriate reviewer for each submitted review.
    - In the **Authors** tab, define the list of developers who are writing code that you want reviewed. For each author, specify an author name and a source control login (if the author's source control login is different than the author's name).
        - Your list of authors can include all of your developers, or only your junior developers.
        - If the developer who commits a code change is not defined in this tab, the change will be marked as coming from an 'undefined author'.
        - If you don't want to define every developer, you can 1) enable the **Filter** tab's **Accept all (also undefined) authors for reviewed paths**, and then 2) Use the **Reviewers** tab to define which reviewers should review different parts of the code.
    - In the **Reviewers** and **Monitors** tabs, specify which authors and/or project areas you want each reviewer or monitor to cover.
        - Reviewers examine, comment on, and approve code changes. Monitors supervise the entire process to ensure that revisions are being reviewed and then corrected in a timely manner. They do not need to perform any reviews, but can add comments to the revisions or reviews. This role is optional.
        - Paths are defined in logical (workspace) path convention. Wildcards are allowed. See the "Filter Tips and Examples" box below for more details and examples.
        - You can define reviewers and monitors without mapping them to any particular path or author. Such users will be not assigned to any package automatically, but they will be included in the report and authors will be able to select them in the Code Review Assistant dialog.
        - These tabs use OR logic (not AND logic). In other words, you define the name of a reviewer (or monitor), then the authors and review paths you want that person to review (or monitor). Then, if new code comes from either the specified authors OR the specified paths, it will be assigned to the named reviewer or monitor.
8. (Optional) In the **Code Review> Filters** tab, modify the following options if desired:
    - **Ignore changes within suppressed blocks:** Enable this option if you want the code review scan to ignore all differences that occur between "codereview-begin-suppress" and "codereview-end-suppress" markers. These are the same markers that are used to prevent the compare editor from displaying differences within specific blocks of code (see Hiding Differences for Specific Pieces of Code for details).
    - **Differences:** If you want the code review scan to ignore all current source vs. previous source differences that match a certain pattern, specify the appropriate regular expression here. For example, you might use this to prevent any differences in automatically-generated comments from being flagged as requiring code review.
    - **Post to Pre-Commit matching:** Contains options for hybrid pre-commit and post-commit code review processes, where developers are expected to commit code for review prior to checking but post-commit scans are also used to validate that this process is being followed.
        - **Post to Pre-Commit matching:** If you want the post-commit scan to ignore all pre-commit scan vs. post-commit scan differences that match a certain pattern, then specify the appropriate regular expression here. For example, you might use this to prevent differences in automatically-generated CVS headers added upon checkin from being flagged for post-commit code review. See Running Post-Commit Scans with a Pre-Commit for more details.
        - **Pre-commit search range in days:** If you want to customize the timeframe used to determine which pre-commit tasks correspond to post-review tasks, change this setting. See Running Post-Commit Scans with a Pre-Commit for more details.
    - **Accept all (also undefined) authors for reviewed path:** If you don't want to define every developer, you can 1) enable the **Accept all (also undefined) authors for reviewed paths**, and then 2) Use the **Reviewers** tab to define which reviewers should review different parts of the code.
9. Click **Apply** to commit the new Test Configuration.
10. Share the Test Configuration by right-clicking it, then choosing **Upload to Team Server** from the shortcut menu.

Submitting Code for Review - Interactive Desktop Execution

Each developer should perform the following steps after completing new/modified code that is ready for review:

1. In the project tree, select the node that represents the changes you want reviewed.
2. Run your designated Code Review Test Configuration. This should be in the Team category.
3. If you enabled the **Show user assistant during scanner run** option in the Code Review Preferences panel (strongly recommended), C++test will display the Code Review assistant.

   You can use this dialog to:
    - Indicate what task the code is related to (this code review will then be associated with this task). You can choose an existing task from the Task list. Or, you can create a new task by entering a new task name. This information is used to determine if you are still working on an existing task, or if you have started a new one. Your changes will be grouped by task in the code review packages that are prepared.
    - Assign a specific reviewer or monitor to the submitted revision when you are submitting a new task for review. This allows you to override the default reviewer/monitor assignment—or to specify a reviewer if one is not already assigned.
    - Provide the reviewer or monitor additional information about the current changes.
    - Indicate the priority of the review.
    - Enter notes related to the selected file(s). These notes will be captured in a "general task." This is an opportunity to add general comments regarding this review package.
4. If your Code Review Test Configuration does not have **Auto publish reviews** enabled, upload the results to Team Server as follows:
    - After the test has completed, click the **Generate Report** button that is available in the Test Progress panel's toolbar.
    - Complete the Report dialog that opens.

The designated reviewer will then be alerted that code is ready for review. The reviewer can perform the review as described in Reviewers - Reviewing Code Modifications.

After the review is completed, the author can respond as described in Authors - Examining and Responding to Review Comments.

Adding New Files to the Code Review Package

To add selected files to an existing code review package:

1. In the project tree, right-click the package to which you want to add files, then choose **Parasoft> Add to> Code Review** from the shortcut menu.
2. Complete the Code Review Assistant dialog.

Defining Reviewers, Authors, and Monitors with a Properties File

In addition to specifying code review users directly in the Test Configuration and adding reviewers "on-the-fly" in the Code Review Assistant dialog, you can also define code review users in a .properties file. This can be useful if your review process involves a large number of team members and you can automatically generate a database of your team members.

To define reviewers with a properties file:

1. Prepare an empty properties file `config.properties`.
2. Add to it the following initial required data:

```
com.parasoft.xtest.checkers.api.config.name=Base Code Review
com.parasoft.xtest.checkers.api.config.tool=3

com.parasoft.xtest.codereview.scanner.checksum=fixed
```

3. Begin defining the list of users—either manually, or using a custom script that pulls the appropriate information automatically. Each user must have an index, and then properties, such as name, roles, and monitored.locations (for monitors only). Available roles are 'a' (author), 'm' (monitor), 'r' (reviewer). For example:

```
com.parasoft.xtest.codereview.scanner.crusers.0.name=tom
com.parasoft.xtest.codereview.scanner.crusers.0.roles=r
com.parasoft.xtest.codereview.scanner.crusers.1.name=bob
com.parasoft.xtest.codereview.scanner.crusers.1.roles=r
com.parasoft.xtest.codereview.scanner.crusers.2.monitored.locations=**
/src/**, **/include/**
com.parasoft.xtest.codereview.scanner.crusers.2.name=joe
com.parasoft.xtest.codereview.scanner.crusers.2.roles=am
com.parasoft.xtest.codereview.scanner.crusers.3.name=bill
com.parasoft.xtest.codereview.scanner.crusers.3.roles=a
```

4. Publish the prepared file on a dedicated HTTP server. If you will be automatically regenerating this based on user database, use a file location that will allow such regeneration.
5. Open the Test Configuration dialog and create new configuration by duplicating the "Pre Commit" Test Configuration.
6. Right-click that Test Configuration, choose **New Child**, then type the URL address to your HTTP parent configuration.
7. Refresh the Test Configuration manager and verify that any added monitors are displayed.
8. Publish it on Team Server.

Here is a sample properties file:

```
com.parasoft.xtest.codereview.scanner.crusers.13.roles=arm
com.parasoft.xtest.codereview.scanner.crusers.13.name=dave
com.parasoft.xtest.codereview.scanner.crusers.13.reviewers=mark_A, mark_B
```

```
com.parasoft.xtest.codereview.scanner.crusers.13.monitors=mike
com.parasoft.xtest.codereview.scanner.crusers.13.reviewed.locations=**/test
/*.txt, project_A/**
com.parasoft.xtest.codereview.scanner.crusers.13.monitored.locations=/**

com.parasoft.xtest.codereview.scanner.checksum=fixed
```

Note that:

- number 13 is a unique id of this user in a config.
- roles "arm" stand for author/reviewer/monitor. The presence of a particular letter indicates that this user has its corresponding role. For example roles "rm" mean that he should be included in list of suggested reviewers/monitors, but he's not an author (no reviews will be created for his revisions).
- reviewers/monitors is a list of users that should be automatically assigned to package authored by this user (requires role author=true).
- reviewed/monitored locations is a list of regions to which this user should be assigned as reviewer/monitor (requires role reviewer /monitor=true).
- reviewer/monitor=true for 'dave' is required also if any other user has 'dave' specified on his reviewers/monitors list
- only unique id and user name is required, but each user has to have at least one of author/reviewer/monitor roles active to be meaningful
- checksum=fixed is a single key required to avoid overwriting of this config by older versions of the application.

Adding Post-Commit Scans To Your Pre-Commit Process

Some teams who submit code for review via a pre-commit process also like to perform a post-commit nightly scan to:

- Generate emails notifying authors and reviewers about their assigned code review tasks.
- Identify any code changes that were committed to source control, but were not submitted for review using the pre-commit procedure.

For details on how to accomplish this, see Running Post-Commit Scans with a Pre-Commit.