Code Review Tips and Tricks

This topic provides tips on using Code Review effectively.

Setup Tips

- To make it easy to associate code review results with the appropriate DTP projects, be sure that your Test Configuration (Common tab> Session Tag) specifies a unique session tag that can be used as a filter when you set up your DTP project. In most cases, we recommend that you 1) Set up your tool's "General Project" preferences option (for details, see Connecting to Project Center) and 2) Set up the Test Configuration Session Tag to also use a project variable.
    - Setting up the "General Project" preferences option and using the `${general_project}` variable works well in most cases. However, if multiple DTP projects are within a single solution or workspace, use `${project_name}` for the Session Tag variable instead of `${general_project}`

      . For more information on variables, see C++test Configuration Overview.
- If your Parasoft solution uses Team Server Named Accounts, ensure that Team Server user accounts can access the Team Server 'Code Review' directory. For details on opening the appropriate path permissions, see the 'Named Accounts' in the user administration section of the DTP documentation.
- Ensure that reviewing responsibilities are distributed across team members. If one person is responsible for reviewing a large amount of code, the code review may become overwhelming.
- If you want to have a single Test Configuration that can be used across multiple Parasoft Test products, see Configuring Cross-Product and Cross-Language Code Review.

Author Tips

- For post-commit code reviews - When you check in code to the source control repository, enter comments that will help the reviewer quickly determine the purpose and nature of the change.

Reviewer Tips

- Review code daily to prevent the reviews from accumulating. If reviews are not addressed regularly, code review will become an overwhelming task.
- For post-commit code reviews - Read the developer's check in comments (available in a tool tip over the revision branch) before reviewing each revision, then let this inform how deeply you review the code. For instance, you might want to dedicate more time to inspecting a revision for a re-implemented algorithm than for inspecting a revision for a minor cosmetic change.