C++test Configuration Overview

This topic explains how you can configure (or update) team-wide settings once, then have those settings propagated across the team's C++test installations. It also covers how to extend or override these settings as needed (for example, for machine-specific paths). Additionally, various C++test options can be specified in the preferences panel.

Sections include:

Workflow Overview

If your organization is using Parasoft Development Testing Platform, we strongly recommend that you adopt the following workflow to simplify configuration and updating of preference settings across your team:

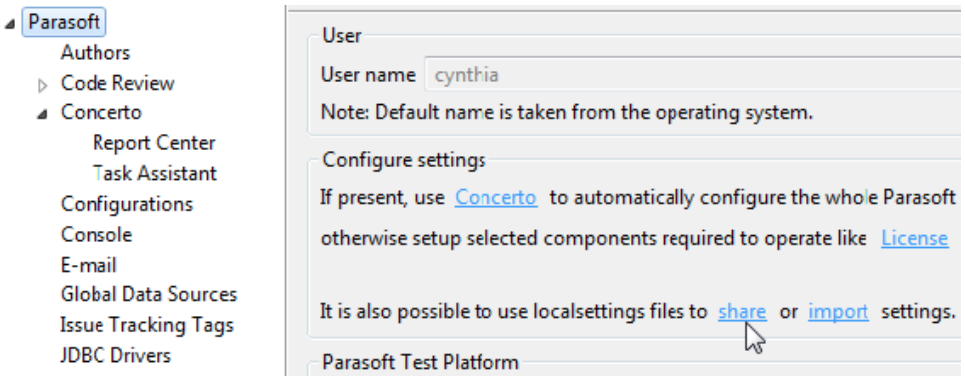
1. Configure team-wide preferences.
    a. Configure team-wide preferences settings in the C++test GUI.
    b. Convert those settings into the localsettings format.
    c. Add the settings to your team's DTP server.
2. Configure C++test preferences on each desktop and server installation.
    a. Let C++test automatically detect the settings stored on DTP.
    b. Extend or override those settings as needed.
3. When team-wide preferences need to be updated, update them on the DTP server. Changes will automatically be propagated to the connected C++test installations.

Configuring Team-wide Preference Settings

Team-wide preference settings can be specified from the GUI, exported into localsettings, then shared via Parasoft DTP.

To configure team-wide settings:

1. Open the Parasoft GUI and configure the following settings:
    - E-mail (described in Configuring Email Settings).
    - License (described in Licensing).
    - Development Testing Platform (described in Connecting to Development Testing Platform
    - Project Center (described in Connecting to Project Center).
    - Team Server (described in Connecting to Team Server).
    - Source control (described in Connecting to Source Control).
    - Authors (described in Specifying Author-to-Author and Author-to-Email Mappings).
    - Any additional settings you want to share. See Preference Configuration Basics for details.
2. In the top-level Parasoft Preferences page, click the **Share** link, specify which settings you want to export, and specify where you want to store the localsettings file that contains the exported preferences.



3. Add the exported settings to Parasoft DTP (edit the project, then paste the settings from the exported localsettings file into the **Parasoft Test Settings** tab).

Configuring Preferences on Each Installation

To configure each desktop and server installation to use the preferences stored on DTP:

1. Open the Parasoft Preferences panel (**Parasoft> Preferences**).
2. Open the **Development Testing Platform** page.
3. Configure your DTP server connection now (see Connecting to Development Testing Platform for details).
4. Update machine-specific settings (for example, settings that involve local paths) as desired—see Preference Configuration Basics for details.
    - If needed, you can change/override imported settings on the related Preferences panel pages (e.g., to override license settings, go to the License page). Just clear the **Use DTP settings** option on the appropriate page, then manually configure the settings.
    - If you do override imported settings and later want to restore them, click the **Restore DTP Defaults** button on the related preference page (to restore specific settings). Or, to reconfigure settings for the selected project and restore all defaults, click the **Restore Defaults** button the Development Testing Platform page.

Working with Multiple DTP Projects

If you're working with multiple DTP projects, you can easily switch from one project's settings to another by clicking the DTP preferences page's **Projects> Configure** button, then changing the selected project.

Refreshing Parasoft Test Settings Specified on DTP

If you are storing C++test settings on DTP, those settings are refreshed each time that C++test is started.

If you want to manually refresh these settings (e.g., if you know that settings changed and you want to retrieve the new settings immediately, without restarting C++test):

1. Choose **Parasoft> Preferences**.
2. Open **Parasoft> Development Testing Platform**.
3. Click the **Configure** button to the right of the **Project** field. This will synchronize the selected project's settings with those on the DTP server.

Using Preference Settings for Command-Line Execution

Using an Existing Locally-Stored Localsettings File

If you already have a locally-stored localsettings file that represents the settings you want to use for command-line execution:

- Specify that file at the command line (e.g., using `-localsettings my_localsettings_file`)

For more details, see Configuring Localsettings and Testing from the Command Line Interface.

Using the Settings Stored on the DTP Server

If you would prefer to use settings stored on the DTP server (recommended for ease of maintenance)—you can do this in either of two ways:

- Specify `-dtp.autoconfig project_name@servername:port` at the command line. For example:
  `-dtp.autoconfig Project1@concerto.company.com:8080`
  This is often used when teams do not already have a locally-stored localsettings file.
- Use the `dtp.autoconfig=true` localsettings option. For example:
  ```
  dtp.enabled=true
  dtp.server=servername
  dtp.port=8080
  dtp.autoconfig=true
  dtp.project=project_name
  ```

  This is often used when a team already has a localsettings file, then wants to import some additional settings from DTP. You can define common properties in DTP, then configure per-test-run specific properties in various localsettings files for different test runs.

For details on how to specify localsettings, see Configuring Localsettings.

Specifying Multiple Groups of Settings

If you want to use a combination of settings (for example, 1) key project settings stored on DTP 2) settings for all tests from this particular machine and 3) settings tailored for just a specific set of analyses from this machine), you can express a hierarchy of files as follows:

1. Export each group of settings as a localsettings file using the procedure described in Exporting GUI Preferences to a localsettings File
2. Use both `-dtp.autoconfig` and `-localsettings` at the command line. For example:
   ```
   -dtp.autoconfig Project1@concerto.company.com:8080 -localsettings
   machine_override_properties -localsettings project_override_properties
   ```
   Be sure to list your most general settings first, and your most specific settings last. Settings will be processed in the order in which they are listed; any settings that are duplicated across groups will be overridden each time a duplicate is found.

Updating Team-wide Settings

If you are using this recommended process, you can updated team-wide settings in Parasoft DTP, then those modifications will automatically be propagated to all the connected machines.

To prevent this automated updating (e.g., because you have updated settings locally and do not want them overridden), disable **Use DTP settings** on the Preferences page(s) you do not want to be updated from DTP.

Preference Configuration Basics

To customize preferences:

1. Choose **Parasoft> Preferences**. The Preferences dialog will open.
2. In the left pane, select the category that represents the settings you want to change. See the following table and the listed references for details.
3. Modify the settings as needed.
4. Click either **Apply** or **OK** to commit the modified settings.

Preferences Categories

Parasoft (Root-Level)

Sets general preferences and allows you to export settings to a localsettings file.

- **Ignore solution and solution folder names in paths:** Enables "reduced paths" mode. In this mode—where only reduced (project relative) paths are used. Names of solution and solution folders will be skipped in these paths. New paths will be unique because project names are unique inside a solution.
- **User name:** Allows you to set a different user name than the one specified in the operating system. C++test uses this user name for code review and Project Center settings. The user name specified here can be overwritten at the code review and Project Center level.
- **Share/import:** See Exporting GUI Preferences to a localsettings File.

Authors

Maps a team member's automatically-detected username to a different username and/or email address.

See Configuring Task Assignment and Code Authorship Settings.

Code Review

Specifies settings for automating the preparation, notification, and tracking of the peer review process.

See Configuring Code Review Preferences.

Configurations

Specifies the number of Test Configurations available in the **Parasoft> Test History** menu, the location where custom static analysis rules (user rules) are saved and searched for, and the location where user-defined Test Configurations and rules are saved and searched for.

- **Size of recently run test configurations:** Determines the number of Test Configurations available in the **Parasoft> Test History** menu.
- **Custom directories:** Indicates where user-defined Test Configurations and custom directories (e.g., for user rules, embedded cross-compilers, etc.) are saved.

Console

Specifies settings for the Console view.

- **Low:** Configures the Console view to show errors and basic information about the current step's name and status (done, failed, up-to-date).
- **Normal:** Adds command lines and issues reported during test and analysis.
- **High:** Uses full-format violation listings and also reports warnings.
- **Show console on any change:** Determines whether the console is brought to the front any time its content changes.

Development Testing Platform

Specifies setting for the DTP server.

See Connecting to Development Testing Platform.

When expanded, it allows you to configure settings for DTP Project Center (see Connecting to Project Center) and Team Server (see Connecting to Team Server) which are deprecated.

E-mail

Specifies email settings used for report notifications and for sending files to Parasoft Technical Support.

See Configuring Email Settings.

Issue Tracking Tags

Specifies custom tags that the team uses to associate a test case with an issue from an issue/feature/defect tracking system (for example, Bugzilla).

See Using Custom Defect/Issue Tracking Tags.

JDBC Drivers

Specifies JDBC drivers (e.g., drivers needed to connect to a database used for parameterizing tests).

See Configuring JDBC Drivers.

License

Specifies license settings.

See Licensing.

Parallel Processing

Specifies parallel processing settings.

See Configuring Parallel Processing.

Quality Tasks

Specifies general options related to how tasks are displayed in the Quality Tasks view.

See Configuring Task Reporting Preferences.

Reports

Specifies what reports include and how they are formatted.

See Configuring Reporting Settings.

Scope and Authorship

Specifies how Parasoft Test computes code authorship and assigns tasks to different team members.

See Configuring Task Assignment and Code Authorship Settings.

Source Control

Specifies how Parasoft Test connects to your source control repositories.

See Connecting Source Control.

Technical support

Specifies options for preparing "support archives" and sending them to the Parasoft support team.

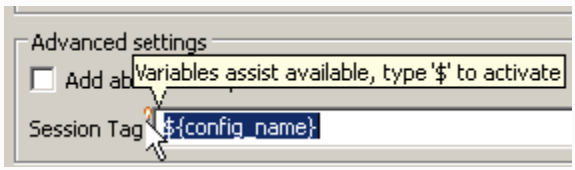
See Preparing a "Support Archive" and Sending it to Technical Support

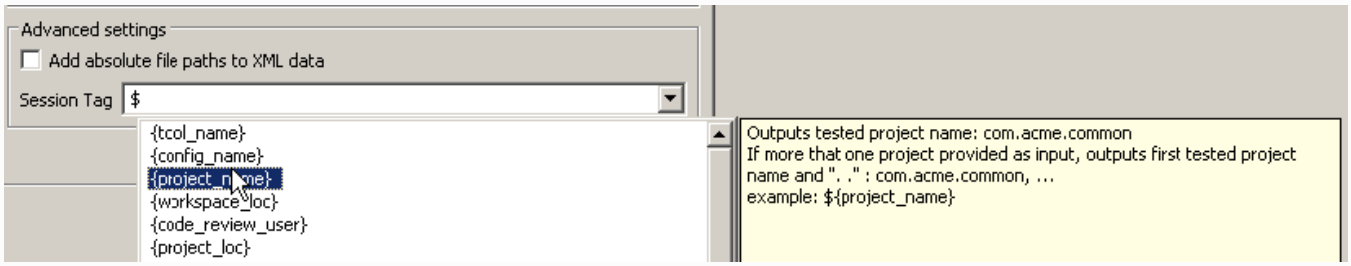
Using Variables in Preference Settings

The following variables can be used in reports, e-mail, Project Center, Team Server, and license settings. Note that the session tag value can't contain any ':' characters.

Using Variable Assist

For help specifying variables, you can use the variable assist feature, which automatically proposes possible variables when you type $. For example:





env_var
example: ${env_var:HOME}
Outputs the value of the environmental variable specified after the colon.

project_name
example: ${project_name}

Outputs the name of the tested project. If more than one project is provided as an input, it first outputs the tested project name, then "..."

general_project
example: ${general_project}

Outputs the name of the DTP general project that results are linked to. See General Project - Notes for details.

workspace_name
example: ${workspace_name}

Outputs the workspace name or Visual Studio solution name. For example,

```
report.mail.subject=Code Review Scanner Results for ${workspace_name}
```

would evaluate to something like "Code Review Scanner Results for solution.sln"

solution_loc (Visual Studio)
example: ${solution_loc}

Outputs the Visual Studio solution location. For example,

```
report.mail.subject=Code Review Scanner Results for ${solution_loc}
```

would evaluate to something like "Code Review Scanner Results for c:/nightly/folder/.../solution.sln"

config_name
example: ${config_name}

Outputs the name of executed test configuration; applies only to Reports and Email settings.

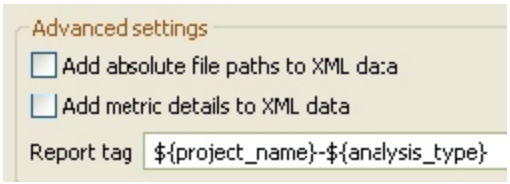
analysis_type
example: ${analysis_type}

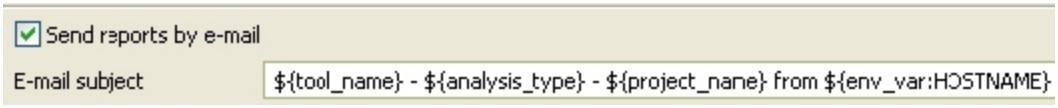
Outputs a comma separated list of enabled analysis types (for example: Static, Execution); applies only to Reports and Email settings.

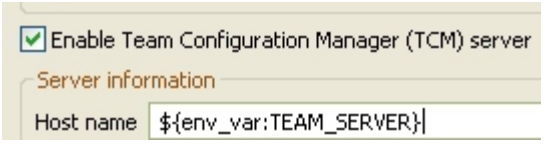
tool_name
$ example: ${tool_name}

Outputs the tool name (for example: C++test, SOAtest).

For example:







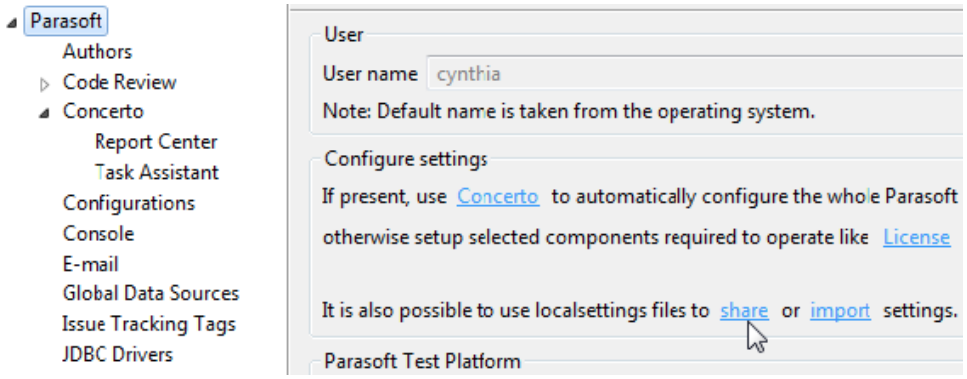
Exporting GUI Preferences to a localsettings File

You can export C++test preferences from the GUI to a localsettings text file. For example, this is useful if you want to:

- Copy settings into DTP for team-wide sharing (see Workflow Overview)
- Quickly create a settings file that you can modify for command-line testing preferences.
- Share the preferences via source control (rather than via DTP).
- Edit preferences in a text file rather than in the GUI.

To export preferences to a localsettings file:

1. Choose **Parasoft> Preferences**. The Preferences dialog will open.
2. Select **Parasoft** (the root element in the left tree) in the Preferences dialog.
3. In the top-level Parasoft Preferences page, click the **Share** link, specify which settings you want to export, then export the settings to a localsettings file.



- If you select an existing file, the source control settings will be appended to that file. Otherwise, a new file will be created.
- Exported passwords will be encrypted.

To import these preferences:

1. Choose **Parasoft> Preferences**. The Preferences dialog will open.
2. Select **Parasoft** (the root element in the left tree) in the Preferences dialog.
3. Click the **Import** link, then specify the file where the saved settings are stored, and which settings you want to import:

Overriding the System User Name Outside of the GUI

If you need to override the system user name (e.g., if you are integrating the product into an automated process and do not want the resulting tasks assigned to the default system name), you can do so as follows:

- Provide the `-Duser.name=<username>` switch to the Java Virtual Machine

Note that this configuration is the equivalent to modifying the **User name** setting at the top level of the Preferences UI.