Configuring Localsettings

This topic explains how you can specify localsettings to control options for reporting, task assignment, licensing, and more. Localsettings can be used to share preferences across a team as well as to apply different groups of settings to different projects and test runs.

Sections include:

About Localsettings

Localsettings can control report settings, Parasoft DTP settings, error authorship settings, Team Server settings, and more. If a parameter is specified in this file and there is an equivalent parameter in the GUI's Preferences panel (available from **Parasoft> Preferences**), the parameter set in this file will override the related parameter specified from the GUI.

Localsettings can be used to:

- Enter GUI-specified and manually-specified settings into Parasoft DTP, which centralizes preference distribution and updating across the team.
- Configure and use different setting configurations for different projects.
- Extend or override team-wide settings as needed (for example, for settings that involve local paths).
- Adjust settings without having to open the GUI.

Defining Localsettings

There are two ways to define localsettings:

- Enter them manually in a simple text file simple text file. There are no name or location requirements. Each local setting should be entered in a single line.
- Export your GUI preferences as described in Exporting GUI Preferences to a localsettings File then adjust or extend them as needed.

Using Variables in Localsettings

For a list of variables that can be used in reports, e-mail, Parasoft Project Center, Team Server, and license settings, see Using Variables in Preference Settings.

Specifying Which Localsettings to Use

Localsettings can be stored on Parasoft DTP (where they are automatically applied to connected Parasoft Test installations) or in a local file (where they can be specified from the command line).

Multiple layers of localsettings can be active for a single test run.

For details on how to store and apply localsettings, see C++test Configuration Overview.

Localsettings Notes

- Each setting should be entered on a single line.
- If a parameter is specified in localsettings, it will override the related parameter specified from the GUI. If a parameter is not specified in localsettings, the parameter specified in the GUI will be used.
- If you are importing preferences from localsettings specified on DTP and you want to override these settings from the GUI, you can clear the **Use DTP settings** option on the appropriate page, then manually configure the settings.
- If any localsettings problems are detected during a test run, details will be reported in the command line output.
- If you are running cli mode from a developer/tester desktop (as opposed to from a Server machine), use the `tasks.clear=false` option to ensure that your results from previous runs are preserved.

Available Settings

Reporting Settings

Setting	Purpose
`build.id`	Specifies a build identifier used to label results. It may be unique for each build but may also label more than one test sessions that were executed during a specified build. Default: build-yyyy-MM-dd HH:mm:ss
`report.active_rules=true|false`	Determines if the reports contain a list of the rules that were enabled for the test. Default: false
`report.archive=true|false`	Enables the generation of an additional compressed archive (.zip) file in the specified report location. The ZIP file contains all the files generated to build the report. This option can generate an archive for any report format (e.g., HTML, CSV, PDF, etc.). By generating an archive, you can also perform custom transformations of the report because all of the elements are generated to the specified destination folder. Default: false
`report.associations`	Specifies whether the report shows requirements, defects, tasks, and feature requests that are associated with a test. Default: false
`report.authors_details`	Determines whether the report includes an overview of the number and type of tasks assigned to each team member. Default: true
`report.contexts_details`	Determines whether the report includes an overview of the files that were checked or executed during testing. Default: false

`report.custom.extension`
 	
`report.custom.xsl.file`
	Specifies the location and extension of the XSL file for a custom format. Used with `report.format=custom`
	
	For details and examples, see Configuring Reporting Settings.

`report.developer_errors=true|false`	Determines whether manager reports include details about team member tasks. Default: false
`report.developer_reports=true|false`	Determines whether the system generates detailed reports for all team members (in addition to a summary report for managers). Default: true
`report.format=html|pdf|sate|xunit|custom`	Specifies the report format. Default: html
`report.generate_htmls=true|false`	Determines whether HTML reports are generated and saved on the local file system. XML reports are generated and saved regardless of this setting's value. Default: true
`report.graph.cs_start_date=[MM/dd/yy]`	Determines the start date for trend graphs that track static analysis tasks over a period of time.
`report.graph.ue_coverage_start_date=[MM/dd/yy]`	Determines the start date for trend graphs that track coverage over a period of time.
`report.graph.ue_start_date=[MM/dd/yy]`	Determines the start date for trend graphs that track test execution results over a period of time.
`report.location_details=true|false`	Specifies whether absolute file paths are added to XML data. This needs to be enabled on the Server installation if you want to relocate tasks upon import to desktop installations. Default: false
`report.mail.attachments=true|false`	Determines whether reports are sent as attachments. All components are included as attachments; before you can view an HTML report with images, all attachments must be saved to the disk. Default: false

`report.mail.cc=[email_addresses]`	Specifies where to mail comprehensive manager reports. This setting must be followed by a semicolon-separated list of email addresses. This setting is typically used to send reports to managers or architects. It can also be used to send comprehensive reports to team members if such reports are not sent automatically (for example, because authorship is not being determined by Parasoft Test).
`report.mail.compact=trends|links`	Specifies that you want to email a compact report or link rather than a complete report. If `trends` is used, the email contains a trend graphs, summary tables, and other compact data; detailed data is not included. If `links` is used, the email contains only a link to a report (which is available on Team Server)
`report.mail.domain=[domain]`	Specifies the mail domain used to send reports.
`report.mail.enabled=true|false`	Determines whether reports are emailed to team members and to the additional recipients specified with the `cc` setting. Remember that each team member with assigned tasks will automatically be sent a report that contains only the assigned tasks. Default: false
`report.mail.exclude=[email_addresses]`	Specifies any email addresses you do not want to receive reports. This set-ting is used to prevent automated sending of reports to someone that worked on the code, but should not be receiving reports.
`report.mail.exclude.developers=true|false`	Specifies whether reports should be mailed to any team member whose email is not explicitly listed in the `report.mail.cc` property. This setting is used to prevent reports from being mailed to individual team members. Default: false
`report.mail.format=html|ascii`	Specifies the email format. Default: html
`report.mail.from=[email_address OR user_name_of_the_same_domain]`	Specifies the "from" line of the emails sent. Default: `<global_user_name>`
`report.mail.include=[email_addresses]`	Specifies the email addresses of team members that you want to receive individual reports. This setting must be followed by a semicolon-separated list of email addresses. This setting is typically used to send individual reports to team members if such reports are not sent automatically (for example, because the team is not using a supported source control system). It overrides team members specified in the 'exclude' list.
`report.mail.on.error.only=true|false`	Determines whether reports are sent to the manager only if a task is generated or a fatal exception occurs. Team member emails are not affected by this setting; individual emails are sent only to team members who are responsible for reported tasks. Default: false
`report.mail.port=[port]`	Specifies the mail server host's port number. Default: 25
`report.mail.security=[SL| STARTTLS| NONE]`	Specifies the desired security. Available settings are SSL, STARTTLS, NONE. SSL is not available in Visual Studio.
`report.mail.server=[server]`	Specifies the mail server used to send reports.
`report.mail.subject=My New Subject`	Specifies the subject line of the emails sent. The default subject line is ${tool_name} Report - ${config_name}. For example, if you want to change the subject line to "Jtest Report for Project A", you would use `report.mail.subject=jtest Report for Project A` Default: `${tool_name} Report - ${config_name}`
`report.mail.time_delay=[server]`	Specifies a time delay between emailing reports (to avoid bulk email restrictions). Default: 0
`report.mail.unknown=[email_address OR user_name_of_the_same_domain]`	Specifies where to mail reports for errors assigned to "unknown".

`report.mail.username=[username] report.mail.password=[password] report.mail.realm=[realm]`	Specifies the settings for SMTP server authentication. The `realm` value is required only for those servers that authenticate using SASL realm.
`report.metrics_details=true|false`	Determines whether an XML report with metrics summary information (as well as individual class and method detail data where applicable) is produced. This report will be generated only when a metrics-enabled Test Configuration is run. Metrics details will be shown in HTML and PDF reports. Default: true
`report.setup.problems=top|bottom|hidden`	Determines whether reports include a section about setup problems. `top` - Adds a "Setup Problems" section to the top of the report. This is the default. `hidden` - Prevents a "Setup Problems" section from being added. `bottom` - Adds a "Setup Problems" section to the bottom of the report. Default: bottom
`report.suppressed_msgs=true|false`	Determines whether reports include suppressed messages. Default: false
`report.test_params=true|false`	Determines whether reports include test parameter details. Default: false
`report.ue_coverage_details_htmls=[coverage_type]`	Determines whether a test's HTML report links to another report that includes source code annotated with line-by-line coverage details. The following values can be used for [coverage_type]: LC - for line coverage SC - for statement coverage (C/C++test only) BCC - for block coverage (C/C++test only) DC - for decision coverage (C/C++test, Jtest only) SCC - for simple condition coverage (C/C++test only) MCDC - for MC/DC coverage (C/C++test only)
`session.tag=[name]`	Specifies a session tag used to label these results. This value is used for uploading summary results to Team Server. The tag is an identifier of the module checked during the analysis process. Reports for different modules should be marked with different tags. Default: `${config_name}`
`tasks.source.control.details=true|false`	This setting specifies if additional information from source control, such as revisions and comments, is included in the report.

Parasoft DTP / Project Center Settings

Setting	Purpose
`dtp.autoconfig=true|false`	Enables autoconfiguration with Parasoft Test settings stored on the DTP server. Default: false
`dtp.enabled=true|false`	Determines whether the current C++testt is connected to DTP. Default: false
`dtp.user=[username]`	Specifies the username for DTP user authentication.
`dtp.password=[password]`	Specifies the password for DTP user authentication.
`concerto.reporting=true|false`	Determines whether the current Parasoft Test product is connected to Parasoft Project Center. Default: false
`dtp.server=[server]`	Specifies the host name of the Parasoft DTP server.

`concerto.data.port=[port]`	Specifies the Parasoft Project Center port. Default: 32323
`dtp.port=[port]`	Specifies the Parasoft DTP server port. Default: 80
`dtp.user_defined_attributes=[attributes]`	Specifies the user-defined attributes for Parasoft Project Center. Use the format `key1:value1; key2:value2` For more details on attributes, see Connecting to Project Center.
`concerto.log_as_nightly=true|false`	Determines whether the results sent to Parasoft Project Center are marked as being from a nightly build. Default: false
`concerto.use_resource_attributes=true|false`	Determines whether Parasoft Project Center attributes specified in the GUI at the project level should be used. This allows you to disable project-level Parasoft Project Center attributes. Default: true
`dtp.project=[project_name]`	Specifies the name of the DTP project that you want these results linked to. For more details on general projects, see Connecting to Project Center. Default: Default Project

Team Server Settings

Setting	Purpose
`tcm.server.enabled=true|false`	Determines whether the current Para-soft Test product is connected to the Parasoft Team Server. Default: false
`tcm.server.name=[name]`	Specifies the machine name or IP address of the machine running Team Server.
`tcm.server.port=[port]`	Specifies the Team Server port number. Default: 18888

`tcm.server.accountLogin=true|false`
 	
`tcm.server.username=[username]`
`tcm.server.password=[password]`
	Determines whether username and password are submitted to connect to Team Server. Usernames/passwords are not always needed; it depends on your team's setup.
	
	If the first setting is `true`, the second and third settings specify the username and password.
	
	Note that Team Server must have the username and password setting already enabled before these settings can be used.
	
	tcm.server.accountLogin default: false

Licensing Settings

See Manually Adding the License to localsetttings for additional notes and examples.

Setting	Purpose
`[product].license.use_network=true|false`	Determines whether the current Parasoft Test product retrieves its license from LicenseServer. Be sure to replace `[product]` with the name for the appropriate Parasoft Test product (for example, `jtest, cpptest, dottest, soatest`). Example: `jtest.license.use_network=true` Default: true
`[product].license. network.host=[host]`	Specifies the machine name or IP address of the machine running LicenseServer Configuration Manager. Example: `cpptest.license.network.host=10.9.1.63`

`[product].license. network.port=[port]`	Specifies the LicenseServer port number. Example: `soatest.license.network.port=2222` Default: 2002

`[product].license. network.edition=[edition_name]`	Specifies the type of license that you want this Parasoft Test product to retrieve from LicenseServer.
	
	`[edition_name]` can be `server_edition`. To use a custom edition, do not set anything after the `"=";` simply leaving the value empty.
	
	Example:
	
	`dottest.license.network.edition=desktop_edition`
	
	`dottest.license.network.edition=server_edition`
	
	Default: custom_edition

`[product].license. autoconf. timeout=[seconds]`	Specifies the maximum number of seconds the Parasoft Test product will wait for the license to be automatically configured from LicenseServer. Default: 20
`[product].license. local.expiration=[expiration]`	Specifies the local license that you want this Parasoft Test product to use. Default: 0
`[product].license. local. password=[password]`	Specifies the local password that you want this Parasoft Test product to use.
`[product].wait.for.tokens.time=[time in minutes]`	Specifies the time that this Parasoft Test product will wait for a license if a license is not currently available. For example to make C++test wait 3 minutes for license tokens, use `cpptest.wait.for.tokens. time=3.` Default: 0

Technical Support Settings

Setting	Purpose
`techsupport. auto_creation=true|false`	Determines whether archives are automatically prepared when testing problems occur. Default: false
`techsupport. send_email=true|false`	Determines whether prepared archives are emailed to Parasoft support. If you enable this, be sure to specify email settings from the GUI or with the options in Reporting Settings. Default: false
`techsupport. archive_location=[directory]`	Specifies where archives are stored.
`techsupport. verbose=true|false`	Determines whether verbose logs are included in the archive. Note that this option cannot be enabled if the logging system has custom configurations. • Verbose logs are stored in the `xtest.log` file within the user-home temporary location (on Windows, this is `<drive>:\Documents and Settings\<user>\Local Settings\Temp\parasoft\xtest`). • Verbose logging state is cross-session persistent (restored on application startup). • The log file is a rolling file: it won't grow over a certain size, and each time it achieves the maximum size, a backup will be created. Default: false
`techsupport.verbose. scontrol=true|false`	Determines whether verbose logs include output from source control commands. Note that the output could include fragments of your source code. Default: false
`techsupport.item. general=true|false`	Determines whether general application logs are included. Default: false

techsupport.item.environment=true|false	Determines whether environment variables, JVM system properties, platform details, additional properties (memory, other) are included in the archive. Default: false
techsupport.advanced=true|false	Specifies if advanced options will be sent. Default: false
techsupport.advanced.options=[option]	Specifies any advanced options that the support team asked you to enter.

Authorship/Scope Settings

Setting	Purpose

authors.mappings.location=team|local|shared	Specifies where the authorship mapping file is stored. This setting defaults to team unless local or shared is specified.
	
	If set to local (recommended), authorship mappings can be set directly in localsettings. See authors.mapping and authors.user{n} for details.
	
	If set to shared, you can store map-pings in a local file using the authors.mappings.file option.
	
	The team and shared options are deprecated. Files specified by these options should be in the previously-used format of:
	
	#author to author
	user1=user3
	user2=user3
	#author to email
	user3=me@parasoft.com
	
	Default: team

authors.mapping{n}=[from_user, to_user]	Specifies a specific author mapping for authors.mappings.location=local, as described above. For example: authors.mappings.location=local authors.mapping1=baduser,gooduser authors.mapping2=brokenuser,fixeduser authors.mapping3=olduser,newuser
authors.user{n}=[username, email, full_name]	Specifies a specific author name and email for authors.mappings.location=local. For example: authors.user1=dan,dan@parasoft.com,Dan Stowe authors.user2=jim,jim@parasoft.com,Jim White
authors.mappings.file=[path]	Specifies the location of a "shared" file as described in authors.mappings.location above. For example: authors.mappings.file=/home/user/dev/temp/author_mapping1.txt
authors.ignore.case=true|false	Determines whether author names are case sensitive. If true, David and david will be considered the same user. If false, David and david will be considered two separate users. Default: false
scope.sourcecontrol=true|false	Determines whether code authorship is computed based on a data from a supported source control system. Default: false
scope.author=true|false	Determines whether code authorship is computed based on Javadoc @author tags. *Jtest only* Default: true
scope.local=true|false	Determines whether code authorship is computed based on the local user. Default: true
scope.recommended.computation=first|random	Determines how Parasoft Test selects the Recommended Tasks for each team member—it can choose n tasks at random (the default) or select the first n tasks reported (n is the maximum number of tasks that Parasoft Test is configured to show each team member per day)

`scope.xmlmap=true|false`	Determines whether task assignment is computed based on XML files that define how you want tasks assigned for particular files or sets of files (these mappings can be specified in the GUI then saved in an XML file). Default: true
`scope.xmlmap.file=[file]`	Specifies the name of the XML file that defines how you want tasks assigned for particular files or sets of files.

Source Control Settings

Defining multiple repositories of the same type

Indexes (numbered from 1 to n) must be added to the prefix if you want to define more than one repository of the same type. For example:

```
scontrol.rep1.type=ccase
scontrol.rep1.ccase.vob=/vobs/myvob1

scontrol.rep2.type=ccase
scontrol.rep2.ccase.vob=/vobs/myvob2
```

If you are defining only one repository, you do not need to use an index. For example:

```
scontrol.rep.type=ccase
scontrol.rep.ccase.vob=/vobs/myvob1
```

AccuRev Repository Definition Properties

Property	Description
`scontrol.rep.type=accurev`	AccuRev repository type identifier.
`scontrol.rep.accurev.host=`	AccuRev server host.
`scontrol.rep.accurev.port=`	AccuRev server port. Default port is 1666.
`scontrol.rep.accurev.login=`	AccuRev user name.
`scontrol.rep.accurev.password=`	AccuRev password.

ClearCase Repository Definition Properties

Property	Description
`scontrol.ccase.exec=`	Path to external client executable (`cleartool`).
`scontrol.rep.type=ccase`	ClearCase repository type name.
`scontrol.rep.ccase.vob=`	Path inside VOB. ccase.vob value + File.separator must be the valid path to a ClearCase controlled directory.

CVS Repository Definition Properties

Property	Description
`scontrol.rep.type=cvs`	CVS repository type identifier.
`scontrol.rep.cvs.root=`	Full CVSROOT value.
`scontrol.rep.cvs.pass=`	Plain or encoded password. The encoded password should be the same as in the `.cvspass` file. For CVS use the value in `.cvspass` from within the user's home directory For CVSNT use the value store in the registry under `HKEY_CURRENT_USER\Software\Cvsnt\cvspass` When you are first logged in to the CVS repository from the command line using "cvs login", the password is saved in the registry. To retrieve it, go to the registry (using regedit), and look for the value under `HKEY_CURRENT_USER->CVSNT> cvspass`. This should display your entire login name (`:pserver:exampleA@exampleB:/exampleC`) encrypted password value.

`scontrol.rep.cvs.useCustomSSHCreden-tials=`	Determines whether the cvs login and password should be used for EXT/SSH connections. Allowed values are `true` and `false`. It is disabled by default.
`scontrol.rep.cvs.ext.server`	If connecting to a CVS server in EXT mode, this specifies which CVS application to start on the server side. Has the same meaning as the CVS_SERVER variable `.cvs` is the default value.
`scontrol.rep.cvs.ssh.loginname=`	Specifies the login for SSH connections (if an external program can be used to provide the login).
`scontrol.rep.cvs.ssh.password=`	Specifies the password for SSH connection.
`scontrol.rep.cvs.ssh.keyfile=`	Specifies the private key file to establish an SSH connection with key authentication.
`scontrol.rep.cvs.ssh.passphrase=`	Specifies the passphrase for SSH connections with the key authentication mechanism.
`scontrol.rep.cvs.useShell=`	Enable an external program (CVS_RSH) to establish a connection to the CVS repository. Allowed values are `true` and `false`. It is disabled by default.
`scontrol.rep.cvs.ext.shell=`	Specifies the path to the executable to be used as the CVS_RSH program. Command line parameters should be specified in the `cvs.ext.params` property.
`scontrol.rep.cvs.ext.params=`	Specifies the parameters to be passed to an external program. The following case-sensitive macro definitions can be used to expand values into command line parameters: • `{host}` repository host • `{port}` port • `{user}` cvs user • `{password}` cvs password • `{extuser}` parameter cvs.ssh.loginname • `{extpassword}` parameter cvs.ssh.password • `{keyfile}` parameter cvs.ssh.keyfile • `{passphrase}` parameter cvs.ssh.passphrase

Git Repository Definition Properties

Property	Description
`scontrol.rep.type=git`	Git repository type identifier.
`scontrol.git.exec=`	Path to Git executable. If not set, assumes git command is on the path.
`scontrol.rep.git.branch=`	The name of the branch that the source control module will use. This can be left blank and the currently checked out branch will be used.
`scontrol.rep.git.url=`	The remote repository URL (e.g., git://hostname/repo.git).
`scontrol.rep.git.workspace=`	The directory containing the local git repository.

Perforce Repository Definition Properties

Property	Description
`scontrol.perforce.exec=`	Path to external client executable (`p4`).
`scontrol.rep.type=perforce`	Perforce repository type identifier.
`scontrol.rep.perforce.host=`	Perforce server host.

`scontrol.rep.perforce.port=`	Perforce server port. Default port is 1666.
`scontrol.rep.perforce.login=`	Perforce user name.
`scontrol.rep.perforce.password=`	Password.
`scontrol.rep.perforce.client=`	The client workspace name, as specified in the P4CLIENT environment variable or its equivalents. The workspace's root dir should be configured for local path (so that files can be downloaded).

Serena Dimensions Repository Definition Properties

Linux Configuration Note

To use Serena Dimensions, Linux users should run Parasoft Test in an environment prepared for using Serena programs, such as 'dmcli'

- LD_LIBRARY_PATH should contain the path to <SERENA Install Dir>/libs.
- DM_HOME should be specified.

Property	Description
`scontrol.rep.type=serena`	Serena Dimensions repository type identifier.
`scontrol.serena.dmroot=`	Path to the Serena Dimensions executable (e.g., `scontrol.serena.dmroot=C\:\\Program Files (x86)\\Serena\\Dimensions 2009 R2\\CM\\`)
`scontrol.rep.serena.login=`	Login name.
`scontrol.rep.serena.password=`	Password.
`scontrol.rep.serena.host=`	Serena Dimensions server host name.
`scontrol.rep.serena.dbname=`	Name of the database for the product you are working with.
`scontrol.rep.serena.dbconn=`	Connection string for that database.
`scontrol.rep.serena.locale=`	The language used (e.g., `scontrol.rep.serena.locale=en_US`).
`scontrol.rep.serena.mapping=`	If the project has been downloaded/moved to a location other than default work area, use this option to specify a mapping between the project (or stream) with the Serena repository and the local project. If you are working in the default work area, you do not need to define mappings.

StarTeam Repository Definition Properties

Property	Description
`scontrol.rep.type=starteam`	StarTeam repository type identifier.
`scontrol.rep.starteam.host=`	StarTeam server host.
`sscontrol.rep.starteam.port=`	StarTeam server port. Default port is 49201.
`scontrol.rep.starteam.login=`	Login name.

`scontrol.rep.starteam.password=`	Password (not encoded).

`scontrol.rep.starteam.path=`	When working with large multi-project repositories, you can improve performance by specifying the project, view, or folder that you are currently working with.
	
	You can indicate either a simple Project name (all views will be scanned when searching for the repository path), a Project/View (only the given view will scanned) or Project/View/Folder (only the specified Star Team folder will be scanned).
	
	Examples:
	
	`scontrol.rep.starteam.path=proj1`
	`scontrol.rep.starteam.path=proj1/view1`
	`scontrol.rep.starteam.path=proj1/view1/folderA`
	`scontrol.rep.starteam.path=proj1/view1/folderA/folderB`

`scontrol.rep.starteam.workdir=`	If the `scontrol.rep.starteam.path` setting specifies a StarTeam view or folder, you can use this field to indicate a new working directory for the selected view's root folder (if the path represents a view) or a new working directory for the selected folder (if the path represents a folder).
	
	Examples:
	
	`scontrol.rep.starteam.workdir=c:\\storage\\dv`
	`scontrol.rep.starteam.workdir=/home/storage/dv`

Subversion Repository Definition Properties

Property	Description
`scontrol.rep.type=svn`	Subversion repository type identifier.
`scontrol.rep.svn.url=`	Subversion URL specifies protocol, server name, port and starting repository path (e.g., svn://buildmachine.foobar.com/home/svn).
`scontrol.rep.svn.login=`	Login name.
`scontrol.rep.svn.password =`	Password (not encoded).
`scontrol.svn.exec=`	Path to external client executable (`svn`).

CM Synergy Repository Definition Properties

Property	Description
`scontrol.rep.type=synergy`	Synergy/CM repository type identifier.
`scontrol.rep.synergy.host=`	Computer on which synergy/cm engine runs. Local host is used when missing. *For Web mode, the host must be a valid Synergy Web URL with protocol and port (e.g., http://synergy.server:8400)*.
`scontrol.rep.synergy.dbpath=`	Absolute synergy database path e.g \\host\db\name (backslash symbols '\' in UNC/Windows paths must be doubled).
`scontrol.rep.synergy.projspec=`	Synergy project spec which contains project name and its version e.g name-version.
`scontrol.rep.synergy.login=`	Synergy user name.
`scontrol.rep.synergy.password=`	Synergy password (not encoded).
`scontrol.rep.synergy.port=`	Synergy port.
`scontrol.rep.synergy.remote_client=`	(UNIX only) Specifies that you want to start ccm as a remote client. Default value is false. Optional. *This is not used for Web mode.*
`scontrol.rep.synergy.local_dbpath=`	Specifies the path name to which your data-base information is copied when you are running a remote client session. If null, then the default location will be used. *This is not used for Web mode.*
`scontrol.synergy.exec=`	Path to external client executable (`ccm`)

Microsoft Team Foundation Server Repository Definition Properties

Property	Description
`scontrol.rep.type=tfs`	TFS repository type identifier.
`scontrol.rep.tfs.url=`	TFS repository URL (for example, `http://localhost:8080/tfs`).
`scontrol.rep.tfs.login =`	TFS user name.
`scontrol.rep.tfs.password=`	TFS password.

Microsoft Visual Source Safe Repository Definition Properties

Property	Description
`scontrol.rep.type=vss`	Visual SourceSafe repository type identifier.
`scontrol.rep.vss.ssdir=`	Path of repository database (backslash symbols '\' in UNC/Windows paths must be doubled).
`scontrol.rep.vss.projpath=`	VSS project path.
`scontrol.rep.vss.login=`	VSS login.
`scontrol.rep.vss.password=`	VSS password.
`scontrol.vss.exec=`	Path to external client executable (`ss`).
`scontrol.vss.lookup=`	Determines whether a full VSS database search is performed to find associations between local paths and repository paths. True or false.

Important Notes

- The repository(n).vss.ssdir property shouldco ntain a UNC value even if the repository database resides locally.
- Be aware of VSS Naming Syntax, Conventions and Limitations. Any character can be used for names or labels, except the following:
    - Dollar sign ($)
    - At sign (@)
    - Angle brackets (< >), brackets ([ ]), braces ({ }), and parentheses (( ))
    - Colon (:) and semicolon (;)
    - Equal sign (=)
    - Caret sign (^)
    - Exclamation point (!)
    - Percent sign (%)
    - Question mark (?)
    - Comma (,)
    - Quotation mark (single or double) (' ")
- VSS 6.0 (build 8163), which is deployed with Visual Studio 6, does not work properly with projects whose names start with a dot (.) symbol. If such a project name is used, subprojects cannot be added.
- Do not use custom working directories for sub-projects (example: Project $/SomeProject has the working directory C:\TEMP\VSS\SomeProject and its subproject $/SomeProject/SomeSubProject has the working directory D:\SomeSubProject).

Miscellaneous Settings

Setting	Purpose
`report.rules=[url_path_to_rules_directory]`	Specifies the directory for rules html files (generated by clicking the `Printable Docs` button in the Test Configuration's Static Analysis tab). For example: `report.rules=file:///C:/Temp/Burt/parasoft/xtest/gendoc/report.rules=../gendoc/` Default: none
`tasks.clear=true|false`	Clears existing tasks upon startup in cli mode. This prevents excessive time being spent "loading existing results." Default: true

`console.verbosity.level=low|normal|high`	Specifies the verbosity level for the Console view. Available settings are: `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`: Also shows command lines and issues reported during test and analysis. `high`: Also shows warnings. Default: low
`[product].custom.rules.dir=[directory]`	Indicates where user-defined rules are saved. Be sure to replace `[product]` with the name for the appropriate Parasoft Test product (for example, `jtest, cpptest, dottest, soatest`).
`[product].custom.configs.dir=[directory]`	Indicates where user-defined Test Configurations are saved. Be sure to replace `[product]` with the name for the appropriate Parasoft Test product (for example, `jtest, cpptest, dottest, soatest`).
`custom.compilers.dir=[directory]`	Overrides the custom compiler directory settings (found in **Parasoft> Configurations> Custom compilers**) and uses the defined directory to search for custom compilers. *C/C++test only*.
`exec.env=[env1; env2; ...]`	Specifies a list of tags that describe the environment where a test session was executed. Tags could describe an operating system (e.g. Windows, Linux), an architecture (e.g. x86, x86_64), a compiler, a browser, etc. These tags describe a complete test session; more environment details could be also added at the test suite, test, or test case levels via the services API.
`issue.tracking.tags=[value]`	Specifies custom issue tracking tags. Multiple tags can be separated by a comma. For example: `issue.tracking.tags=@custom,@pr ,@fr` For more details, see Indicating Code and Test Correlations.
`parallel.mode=Manual|Auto|Disabled`	Determines which of the following modes is active: • **Auto**: Allows Parasoft Test to control parallel processing settings. • **Manual**: Allows you to manually configure parallel processing settings to suit your specific needs. • **Disabled**: Configures Parasoft Test to use only one of the available CPUs. For more details on this and other parallel processing options, see Configuring Parallel Processing. Default: Auto
`parallel.max_threads=<number>`	Specifies the maximum number of parallel threads that can be executed simultaneously. The actual number of parallel threads is determined based on the number of CPUs, available memory, and license settings. Default: [available_processors]
`parallel.free_memory_limit=<percentage>`	Specifies the amount of memory that should be kept free in low memory conditions (expressed as a percentage of the total memory available for the application). This is used to ensure that free memory is available for other processes. Default: 25
`parallel.no_memory_limit=true|false`	Indicates that you do not want to place any restrictions (beyond existing system limitations) on the memory available to Parasoft Test. Default: false

Additional Options for C/C++test

C/C++test Project Creation and Import Settings

Note that any C/C++test options for creating or importing projects are valid *only when creating or importing the project*. They are ignored during subsequent runs.

Settings for Creating BDF-Based Projects

Option	Description

`bdf.import.location=[WORKSPACE|BDF_LOC|<path>]`	You can specify an external location, or use the keyword WORKSPACE. If WORKSPACE is used, projects will be created in subdirectories within the workspace directory. If BDF_LOC is used and one project will be created, then it will be created in the exact location as the bdf file. If more then one project will be created, then the projects will be created in subdirectories within the bdf file location. Those subdirectories will be named with corresponding projects names. If an external path is specified, then the project will be created in the specified location. WORKSPACE is the default.
`bdf.import.pathvar.enabled=[true|false]`	Specifies if Path Variables should be used in linked folders that will be created in the new projects. The default is false.
`bdf.import.pathvar.name=<name>`	Specifies the name of the Path Variable (if Path Variables are used, per the bdf.import.pathvar.enabled property). The default Path Variable name is DEVEL_ROOT_DIR.
`bdf.import.pathvar.value=<path>`	Specifies the value of the Path Variable (if Path Variables are used, per the bdf.import.pathvar.enabled property). The default value is the most common root directory for all linked folders.
`bdf.import.compiler.family=<compiler_family>`	Specifies what compiler family will be used (for example, `vc_6_0, vc_7_0, vc_7_1, vc_8_0, gcc_2_9, gcc_3_2, gcc_3_3, gcc_3_4, ghs_4_0`). For a custom compiler, you need to use the custom compiler family identifier, which is the name of the directory containing gui.properties, c.psrc and cpp.psrc files). If this property is not specified, the default values will be used.
`bdf.import.c.compiler.exec=<exec>`	Specifies the executable of the C compiler that will be used in the created project.
`bdf.import.cpp.compiler.exec=<exec>`	Specifies the executable of the C++ compiler that will be used in the created project.
`bdf.import.linker.exec=<exec>`	Specifies the executable of the linker that will be used in the created project.
`bdf.import.project.<proj_name>=dir1;dir2;dir3`	Specifies the set of folders to link for the project prj_name. Folders should be specified as a value list of folder paths, separated with semicolons.

Settings for Importing Green Hills .gpj Projects

Setting	Purpose
`gpj.import.location=WORKSPACE|ORIG|<path>`	Specifies the location of the imported projects. If WORKSPACE is used, then the project will be created in workspace. If ORIG is used, then the project will be created in the .gpj project location. If an external path is specified, then the project will be created in the specified location. The default value is WORKSPACE.
`gpj.import.linked=true|false`	Specifies whether the .gpj project source folders are linked into the created Eclipse project. The default value is true.
`gpj.import.subdirs=true|false`	Applicable when `gpj.import.location=<path>` Specifies whether the project(s) are imported into subdirectories or directly into the specified location. If you want the project(s) imported into subdirectories created in the specified external location, use true. If you are importing only one project and you want it imported directly into the specified external location, use false. The default value is true (subfolders are created for each project imported into in external location).

`gpj.import.pathvar.enabled=true|false`	Specifies if path variables should be used when creating linked directories (if the above option is set to true). The default value is false.
`gpj.import.pathvar.name=<name>`	Specifies the path variable name. The default value will be used unless you specify a path variable name that points to a different location (for instance, DEVEL_ROOT_DIR). If a project with the specified name is already defined in the Eclipse workspace and it points to a different location than the value passed in the `gpj.import.pathvar.location` property, then Path Variable will not be used; full paths will be used instead. Also the default value of `gpjimport.pathvar.name` will always be DEVEL_ROOT_DIR if the `gpjimport.pathvar.name` property is not specified. If this property is specified with some <name>, then that <name> will be used as the Path Variable name. The default value is DEVEL_ROOT_DIR.
`gpj.import.pathvar.value=<path>`	Specifies the path variable value. By default, C++test calculates the common root for all linked folders.
`gpj.import.compiler.family=name`	Specifies the compiler family (compiler ID)
`gpj.import.c.compiler.exec=name`	Specifies the C compiler executable
`gpj.import.cpp.compiler.exec=name`	Specifies the C++ compiler executable
`gpj.import.linker.exec=name`	Specifies the linker executable

Settings for Importing Microsoft Visual Studio 6.0 .dsp Projects

Setting	Purpose
`dsp.import.location=WORKSPACE|DSP_LOC|<path>`	Specifies the location of the imported projects. If WORKSPACE is used, then the project will be created in workspace. If DSP_LOC is used, then the project will be created in the .dsp project location. If an external path is specified, then the project will be created in the specified location. The default value is WORKSPACE.
`dsp.import.linked=true|false`	Specifies whether the .dsp project source folders are linked into the created Eclipse project. The default value is true.
`dsp.import.subdirs=true|false`	Applicable when `dsp.import.location=<path>` Specifies whether the project(s) are imported into subdirectories or directly into the specified location. If you want the project(s) imported into subdirectories created in the specified external location, use true. If you are importing only one project and you want it imported directly into the specified external location, use false. The default value is true (subfolders are created for each project imported into in external location).

`dsp.import.pathvar.enabled=true|false`	Specifies if path variables should be used when creating linked directories (if the above option is set to true). The default value is false.
`dsp.import.pathvar.name=<name>`	Specifies the path variable name. The default value will be used unless you specify a path variable name that points to a different location (for instance, DEVEL_ROOT_DIR). If a project with the specified name is already defined in the Eclipse work-space and it points to a different location than the value passed in the `dsp.import.pathvar.location` property, then Path Variable will not be used; full paths will be used instead. Also the default value of `dsp.import.pathvar.name` will always be DEVEL_ROOT_DIR if the `dsp.import.pathvar.name` property is not specified. If this property is specified with some `<name>`, then that `<name>` will be used as the Path Variable name. The default value is DEVEL_ROOT_DIR.
`dsp.import.pathvar.location=<loc>`	Specifies what location the path variable points to. By default, the automatically-generated location will be used. This location is the common root location for all linked directories. If it is not possible to find a common location (for example because .dsp projects are on multiple drives) or the specified location cannot be used, then the path variable will not be used. Full paths will be used instead. For example, assume you have the following paths: path1: c:\a\b\c\proj1 path2: c:\a\b\proj2 The common root location would be c:\a\b The default value is automatically-generated.
`dsp.import.config=<name>`	Specifies which .dsp project configuration should be used. If the specified configuration cannot be found in the imported project, then the default configuration will be used. The configuration name can be passed in two ways: <project_name> - <configuration_name> or only <configuration_name>. If more then one project is imported, then only <configuration_name> should be entered. This prompts the wizard to search for that configuration in all projects. The default value is the default from .dsp.

For example, if the folder C:\temp\sources should be linked in an imported project and we have defined the path variable DEVEL_ROOT_DIR with the value C:\temp, then that folder will be linked as DEVEL_ROOT_DIR/sources and the DEVEL_ROOT_DIR path variable will be created in the workspace. If such a variable cannot be used (for example, because its value points to another folder not containing C:\temp\sources folder, it is already defined and has different value, or it has an invalid value), then C:\temp\sources folder will be linked using the full path C:\temp\sources.

Settings for Importing Keil uVision Projects

Setting	Purpose
`uv.import.location=WORKSPACE|ORIG|<path>`	Specifies the location of the imported projects. If `WORKSPACE` is used, then the project will be created in workspace. If `ORIG` is used, then the project will be created in the original project file location. If an external path is specified, then the project will be created in the specified location. The default value is `WORKSPACE`.
`uv.import.linked=true|false`	Specifies whether the uVision project source folders are linked into the created Eclipse project. The default value is `true`.
`uv.import.subdirs=true|false`	Applicable when `uv.import.location=<path>` Specifies whether the project(s) are imported into subdirectories or directly into the specified location. If you want the project(s) imported into subdirectories created in the specified external location, use `true`. If you are importing only one project and you want it imported directly into the specified external location, use `false`. The default value is true (subfolders are created for each project imported into in external location).
`uv.import.pathvar.enabled=true|false`	Specifies if path variables should be used when creating linked directories (if the above option is set to true). The default value is false.

`uv.import.pathvar.name=<name>`	Specifies the path variable name. The default value will be used unless you specify a path variable name that points to a different location (for instance, `DEVEL_ROOT_DIR`). If a project with the specified name is already defined in the Eclipse workspace and it points to a different location than the value passed in the `uv.import.pathvar.location property`, then Path Variable will not be used; full paths will be used instead. Also, the default value of `uv.import.pathvar.name` will always be `DEVEL_ROOT_DIR` if the `uv.import.pathvar.name` property is not specified. If this property is specified with a `<name>`, then that `<name>` will be used as the Path Variable name. The default value is `DEVEL_ROOT_DIR`.
`uv.import.pathvar.value=<path>`	Specifies the path variable value. By default, C++test calculates the common root for all linked folders.
`uv.import.config=<name>`	Specifies the name of the build configuration to use.

Settings for Importing Renesas High-performance Embedded Projects

Setting	Purpose
`hew.import.location=WORKSPACE|ORIG|<path>`	Specifies the location of the imported projects. If `WORKSPACE` is used, then the project will be created in workspace. If `ORIG` is used, then the project will be created in the original project file location. If an external path is specified, then the project will be created in the specified location. The default value is `WORKSPACE`.
`hew.import.linked=true|false`	Specifies whether the HEW project source folders are linked into the created Eclipse project. The default value is `true`.
`hew.import.subdirs=true|false`	Applicable when `hew.import.location=<path>` Specifies whether the project(s) are imported into subdirectories or directly into the specified location. If you want the project(s) imported into subdirectories created in the specified external location, use `true`. If you are importing only one project and you want it imported directly into the specified external location, use `false`. The default value is true (subfolders are created for each project imported into in external location).
`hew.import.pathvar.enabled=true|false`	Specifies if path variables should be used when creating linked directories (if the above option is set to true). The default value is `false`.
`hew.import.pathvar.name=<name>`	Specifies the path variable name. The default value will be used unless you specify a path variable name that points to a different location (for instance, `DEVEL_ROOT_DIR`). If a project with the specified name is already defined in the Eclipse workspace and it points to a different location than the value passed in the `hew.import.pathvar.location` property, then Path Variable will not be used; full paths will be used instead. Also the default value of `hew.import.pathvar.name` will always be `DEVEL_ROOT_DIR` if the `hew.import.pathvar.name` property is not specified. If this property is specified with a `<name>`, then that `<name>` will be used as the Path Variable name. The default value is `DEVEL_ROOT_DIR`.
`hew.import.pathvar.value=<path>`	Specifies the path variable value. By default, C++test calculates the coomon root for all linked folders.
`hew.import.config=<name>`	Specifies the name of the build configuration to use.

Sample Localsettings

Example 1

```
# Team Server settings: (these may be redundant with settings already specified in Team Preferences of the
installed version, so may not be needed).
tcm.server.enabled=true
tcm.server.name=<team_server.company.com>

# Report settings
report.developer_errors=true
report.developer_reports=true
report.format=html
session.tag=<project name>

# Mail settings:
report.mail.enabled=true
report.mail.cc=<manager1@mailserver.com1;manager2@mailserver.com1>
report.mail.server=mail.company.com
report.mail.domain=company.com
report.mail.subject=<Static Analysis results on Project X>
report.mail.attachments=true
```

Example 2

```
  # Team Server settings
  tcm.server.enabled=true
  tcm.server.name=teamserver.mycompany.com   tcm.server.port=18888
  tcm.server.accountLogin=true
  tcm.server.username=tcm_user
  tcm.server.password=tcm_pass

  # Parasoft Project Center settings
  concerto.reporting=true
  dtp.server=concerto.mycompany.com
  dtp.port=32323

  # Mail settings
  report.mail.enabled=true
  report.mail.server=mail.mycompany.com
  report.mail.domain=mycompany.com
  report.mail.cc=project_manager
  report.mail.subject=Coding Standards
  concerto.log_as_nightly=true
```

Example 3

```
# REPORTS

#Determines whether reports are emailed to developers and to the additional recipients specified with the cc
setting.
#Remember that if the team is using CVS for source control and each developer's email address matches his or
her CVS username + the mail domain, each developer that worked on project code will automatically be sent a
report that contains only the errors/results related to his or her work.

report.mail.enabled=true

#Exclude developers emails (true/false)
report.mail.exclude.developers=false

# Append developers errors to manager emails (true/false)
report.developer_errors=true

# Send reports to developers (true|false)
```

```
report.developer_reports=true

# Append suppressed messages (true|false)
report.suppressed_msgs=false

#Determines where to mail complete test reports.
#This setting is typically used to send reports to managers or architects.
#It can also be used to send reports to developers if developer reports
#are not sent automatically (for example, because the team is not using CVS).
report.mail.cc=manager@domain.com; ${env_var:USERNAME} @domain.com

# mail target for unknown developer errors
report.mail.unknown=manager@domain.com

#Specifies the mail server used to send reports.
report.mail.server=mail_server.domain.com

#Specifies the mail domain used to send reports.
report.mail.domain=domain.com

#Specify mali from
report.mail.from=nightly

#Specifies any email addresses you do not want to receive reports.
#This setting is used to prevent from automatically sending reports to someone that worked on the code, but
should not be receiving reports. This setting is only applicable if the team is using CVS for source control
and developer reports are being sent automatically.
report.mail.exclude=developer1;developer2

# Specifies the subject line of the emails sent.
report.mail.subject= ${tool_name} Report - ${config_name}

# Report test params include (true|false)
report.test_params=true

# Team Server

#Determines whether the current installation is connected to the Team Server.
tcm.server.enabled=true

#Specifies the machine name or IP address of the machine running Team Server.
tcm.server.name=team_server.domain.com

#Specifies the Team Server port number.
tcm.server.port=18888

tcm.server.accountLogin=true
tcm.server.username=user
tcm.server.password=password
session.tag= ${config_name}

# Parasoft Project Center

#Determines the current installation is connected to Parasoft Project Center.
concerto.reporting=true

#Specifies the host name of the Parasoft Project Center server.
dtp.server=grs_server.domain.com

# Specifies the port number of the Parasoft Project Center report collector.
concerto.data.port=32323

# Specifies user-defined attributes for Parasoft Project Center.
#Use the format key1:value1; key2:value2
#Attributes help you mark results in ways that are meaningful to your organization.
#They also determine how results are grouped in Parasoft Project Center and how you can filter results in
Parasoft Project Center.
#For example, you might want to label results by project name and/or by project component name. #Each attribute
contains two components: a general attribute category name
#and a specific identification value. For example, assume your organization wants to classify results by
project.
```

```
#You might then use the attribute project:projname1. For the next project, you could use a different
#localsettings file that specified an attribute such as project:projname2.
dtp.user_defined_attributes=Type:Nightly;Project:Project1

# Determines whether the results sent to Parasoft Project Center are marked as being from a nightly build.
DTP.log_as_nightly=true

# SCOPE

#code authorship based on CVS
scope.sourcecontrol=true

#code authorship based on author tag
scope.author=false

#code authorship based on local user
scope.local=false

# LICENSE

#override license settings
#jtest.license.autoconf.timeout=40
jtest.license.use_network=true
jtest.license.network.host=license_server.domain.com
jtest.license.network.port=2222
jtest.license.network.edition=server_edition

# SOURCE CONTROL

scontrol.rep1.type=cvs
scontrol.rep1.cvs.root=:pserver:developer@cvs_server.domain.com:/home/cvs/scontrol.rep1.cvs.pass=mypassword
```