CLI Options

Available `soatestcli` options are listed in the following tables.

Double vs. Single Quotation Marks

Use "double-quotes" (not 'single quotes') to specify options. For example: `-config "team://Our Configuration"`

Option	Purpose	Notes
`-appconsole stdout`	Prints verbose output to stdout.	This prints the same type of output that would be shown in the console (if running from the GUI).
`-data %WORKSPACE_DIR%`	Specifies the location of the Eclipse workspace directory to use.	Defaults to the current user's dependent directory. If the -data option is not used, then the default workspace found under [SOAtest_workspace]\parasoft\workspace (where "SOAtest_workspace" could be C:\Users\yourname) will be used.

`-import %ECLIPSE PROJECT%`	Imports the specified Eclipse project(s) into the Eclipse workspace.	If %ECLIPSE_PROJECT% is a .project file, the selected project will be imported
		If it is a directory, all Eclipse projects found in the selected directory and subdirectories will be imported.
		
		Examples:
		`-import \".project\"`
		`-import \"c:\\DevelRootDir\"`
		
		If needed, run `soatestcli` with `-import` multiple times: once for each project or set of projects you want to import. Once you have imported all necessary projects, you run `soatestcli` without `-import`—e.g., with with `-config` and any other desired arguments to execute tests.

`-resource %RESOURCE%`	Specifies the path to the test suite(s) to run.	To run a single test suite, specify the path to <test suite name.tst> relative to the workspace.
		
		To run all test suites within a directory, specify the directory path relative to the workspace.
		
		Use multiple times to specify multiple resources. Use quotes when the resource path contains spaces or other non-alphanumeric characters.
		
		If `%RESOURCE%` is a `.properties` file, the value corresponding to `com.parasoft.xtest.checkers.resources` will be interpreted as a colon(:)-separated list of resources. Only one properties file can be specified in this way. If `%RESOURCE%` is a `.lst` file, each line will be treated as a resource. If no resources are specified on the command line, the complete workspace will be tested.
		
		Team Project Set File (PSF) files are supported for CVS, SVN, Star Team, and other source control systems (depending on the Eclipse plugin capabilities installed).
		
		Paths (even absolute ones) are relative to the workspace specified by the `-data` parameter.
		
		If you are specifying multiple tests from different projects, note that tests will be grouped project-by-project, in the order specified in the multiple -resource parameters or in the .lst file. All tests in the same project as the first resource will be run before any tests that are in a different project (e.g., if you specify resources in the order /ProjectA/A.tst, /ProjectB/B.tst, /ProjectA/C.tst, they will executed in the order /ProjectA/A.tst, /ProjectA/C.tst, /ProjectB/B.tst).
		
		Examples:
		`-resource "Acme Project"`
		`-resource "/MyProject/tests/acme"`
		`-resource testedprojects.properties`

`-config %CONFIG_URL%`	Specifies that you want to run the Test Configuration available at `%CONFIG_URL%`.	This parameter is required.
		
		`%CONFIG_URL%` is interpreted as a URL, the name of a Test Configuration, or the path to a local file.
		
		Examples:
		
		• By filename:
		`-config "mylocalconfig.properties"`
		• By URL:
		`-config "http://intranet.acme.com/SOAtest/team_config.properties"`
		• Built-in configurations:
		`-config "builtin://Demo Configuration"`
		`-config "Demo Configuration"`
		• User-defined configurations:
		`-config "user://My First Configuration"`
		• Team configurations:
		`-config "team://Team Configuration"`
		`-config "team://teamconfig.properties"`

`-localsettings %LOCALSETTINGS_FILE%`	Reads the local setting file `%LOCALSETTINGS_FILE%` for global preferences. These settings specify details such as Report Center settings, email settings, and Team Server settings.	The local setting file is a properties file. These files can control reporting preferences (who should reports be sent to, how should those reports be labelled, what mail server and domain should be used, etc.) Team Server settings, Report Center settings, email settings, and more. For details on creating local setting files, see Local Settings Files - Options.
`-publish`	Publishes the reports to the DTP.	In SOAtest 9.10.2 and later, this option sends report data to DTP (requires DTP 5.3.x or later). In older versions of SOAtest, this option sent reports to Team Server.
`-publishteamserver`	Publishes the reports to the Team Server.	The Team Server location can be specified in the GUI or in the local setting file (described in the `-localsettings %LOCALSETTINGS_FILE%` entry).
`-report %REPORT_FILE%`	Generates an XML report to the given file `%REPORT_FILE%` and adds an HTML (or PDF or custom format—if specified using the `report.format` option) report with the same name—and a different extension—in the same directory.	All of the following commands will produce an HTML report `filename.html` and an XML report `filename.xml`. • `-report filename.xml` • `-report filename.htm` • `-report filename.html` If the specified path ends with an ".html"/".htm"/ ".xml" extension, it will be treated as a path to the report file to generate. Otherwise, it will be treated as a path to a directory where reports should be generated. If the file name is explicitly specified in the command and a file with this name already exists in the specified location, the previous report will be overwritten. If your command doesn't explicitly specify a file name, the existing report file will not be overwritten—the new file will be named repXXXX.html, where XXXX is a random number. If the `-report` option is not specified, reports will be generated with the default names "report.xml/html" in the current directory.

`-router matchWhole <searchURI:URI> <replaceURI:URI>`	Specifies search and replace arguments	For example:
		`-router searchURI:host1.adobe.com replaceURI:host2.adobe.com`
		OR
		`-router searchURI:* replaceURI:http://host2.adobe.com/service`
		
		This feature is now deprecated. Please use Environments instead.

`-testName [match:] <test name>`	Specifies test name patterns; test suite names are valid	Allows you to specify the name of the test in the test suite to run. For example, if you want to run a test suite named WSDL Tests, you might use `soatestcli.exe -data "C:\workspace" -resource "MyService" -config "user:// "Example Configuration" -testName "WSDL Tests"` SOAtest will find tests that contain the specific string specified, but it does not perform actual pattern matching (such as wildcards or Regular Expressions). For example, `-testName match: something` will run all tests whose names contain the word `something`. To run multiple tests use `-testName name1 -testName name2` where `name1` and `name2` correspond to the names of the desired tests. Note that you can surround the value with quotes in order to allow spaces in the name. For example, `-testName match: "hello world"` will search for a test with the exact string `hello world` in its name. To limit row usage for tests matching the specified name, you can use `dataSourceRow:<row>` and `dataSourceName: <name>` parameters immediately following `[match:] <test name>`. `dataSourceRow:` can take a list of row numbers or row ranges. For example: 5 1,2,5 3-9 2-5,7,20-30 The `dataSourceName: <name>` argument is optional. if used, it must come after `dataSourceRow:<row>`
`-dataSourceRow <row> -dataSource-Name <name>`	Runs all tests with the specified data source row(s)	The `-dataSourceName <name>` argument is optional. if used, it must come after `-dataSourceRow <row>`. `dataSourceRow <row>` can take a list of row numbers or row ranges. For example: 5 1,2,5 3-9 2-5,7,20-30 For example: • `-dataSourceRow 1-5` will cause any test that is using a data source to run with rows 1 to 5. • `-dataSourceRow 1,5 -dataSource-Name "Data"` will cause any test that is using a data source named "Data" to run with row 1 and row 5. If you want to force all data source rows to be used—even if the data sources were saved to use only specific rows— use `-dataSourceRow all`.
`-dataGroupConfig`	Specifies the active data source within a data group	This argument must be followed by the location of an XML file that specifies the active data source for each data group within each .tst file contained in the test run. The file should be formatted as shown in datagroupConfig XML File Format.
`-environment <environment_name>`	Specifies environment options	When running functional tests from the command line, you can override the active environment specified in a project with one specified from the command line. Note that if the specified environment is not found in the project, the default active environment will be used instead.
`-environmentConfig`	Specifies the active environment variables	This argument must be followed by the location of an XML file that specifies the environment variable values to use for each .tst file contained in the test run. The file should be formatted as shown in environmentConfig XML File Format.
`-fail`	Fails the build by returning a non-zero exit code if any violations are reported.	The return code will be 2 for static analysis violations, 4 for functional test violations, 8 for code review violations, and 1 for any other problem. Also see CLI Exit Codes.
`-Centrasite`	Report test results to the Software AG CentraSite Active SOA registry	Allows you to send results back to the Software AG CentraSite Active SOA registry. For details, see Using Software AG CentraSite Active SOA with SOAtest.

`-qualityCenter` `-qualityCenterReportAllTraffic`	Report test results to HP Quality Center	Allows you to send results back to HP Quality Center. For details, see Using HP ALM and HP Quality Center with SOAtest.
`-testManager` `-testManagerVerbose`	Report test results to Rational TestManager	Allows you to send results back to Rational TestManager. For details, see Using IBM /Rational with SOAtest. Verbose mode provides more information such as request and response traffic
`-visualStudio`	Report test results to Microsoft Visual Studio Team System	Allows you to send results back to Microsoft VIsual Studio Team System. For details, see Using Microsoft with SOAtest.
`-include %PATTERN% -exclude %PATTERN%`	Specifies files to be included/excluded during testing.	You must specify a file name or path after this option. Patterns specify file names, with the wildcards * and ? accepted, and the special wildcard ** used to specify one or more path name seg-ments. Syntax for the patterns is similar to that of Ant filesets. Examples: -include **/Bank.xml (test Bank.xml files) -include **/ATM/Bank/*.xml (test all .xml files in folder ATM/Bank) -include c:/ATM/Bank/Bank.xml (test only the c:/ATM/Bank/Bank.xml file) -exclude **/internal/** (test everything except classes that have path with folder "internal") -exclude **/*Test.xml (test everything, but files that end with Test.xml) Additionally if a pattern is a file with a .lst extension, it is treated as a file with a list of patterns. For example, if you use -include c:/include.lst and include.lst contains the following (each line is treated as single pattern): **/Bank.xml **/ATM/Bank/*.xml c:/ATM/Bank/Bank.xml then it has same effect as specifying: -include **/Bank.xml -include **/ATM/Bank/*.xml -include c:/ATM/Bank/Bank.xml"
`-browserTestsVisible`	For browser tests, opens the browser UI and plays back tests in the browser.	This gives you the option to view and capture the browser contents shown after each test step (e.g., for compliance purposes).
`-concerto.autoconfig %PROJECT_NAME@SERVER_NAME:port%`	Pulls settings stored on the Concerto server (recommended for ease of maintenance—especially if you do not already have a locally-stored localsettings file)	For example: `-concerto.autoconfig Project1@concerto.company.com:8080`
`-encodepass <plain password>`	Generates an encoded version of a given password.	Prints the message 'Encrypted password: <encpass>' and terminates the cli app. Must be used along with `-config <url>.`
`-showdetails`	Prints detailed test progress information.	N/A
`-J`	Specifies additional JVM options, which in turn get passed to the Eclipse executable via the -vmargs option.	The Eclipse -vmargs argument is used to customize the operation of the Java VM to use to run Eclipse. If specified, this option must come at the end of the command line. Even if not specified on the executable command line, the executable will automatically add the relevant arguments (including the class being launched) to the command line passed into Java using the -vmargs argument. Java Main then stores this value in eclipse.vmargs. Usage is `-vmargs [vmargs*] (Executable, Main)`

`-prefs %PREFS_URL%`	Reads the `%PREFS_URL%` preference URL to import Eclipse workspace preferences.	`%PREFS_URL%` is interpreted as a URL or the path to a local Eclipse workspace preferences file. The best way to create a workspace preferences file is to use the Export wizard. To do this:
		
		1. Choose **File> Export**.
		2. In the Export Wizard, select **Preferences**, then click **Next**.
		3. Do one of the following:
		• To add all of the preferences to the file, select **Export all**.
		• To add only specified preferences to the file, select **Choose specific preferences to export**, then check the preferences you want to import.
		4. Click **Browse...** then indicate where you want the preferences file saved.
		5. Click **Finish**.
		
		We recommend that you delete non-applicable properties and keep only critical properties, such as the classpath property. We also recommend that you replace machine/user-specific locations with variables by using the `$(VAR)` notation. These variables will be replaced with the corresponding Java properties, which that can be set at runtime by running `soatestcli` with `-J-D` options (for example `soatestcli -J-DHOME=/home/user`).
		
		Examples:
		`-prefs "http://intranet.acme.com/SOAtest/workspace.properties"`
		`-prefs "workspace.properties"`

`-help`	Displays help information.	Does not run testing.
`-version`	Displays version number.	Does not run testing.
`-initjython, -installcertifi-cate, -unin-stallcertificate`	Installer options	N/A
`-machineid`	Prints the machine ID.	The machine ID is used for licensing purposes.

Notes

- To see a list of valid command line options, enter for `soatestcli -help`.
- `soatestcli` automatically emails designated group managers and architects a report that lists all team/project tasks and identifies which team member is responsible for each task. If no tasks are reported, reports will be sent unless the local setting file contains the `report.mail.on.error.only=true` option.
- If the appropriate prerequisites are met, `soatestcli` automatically emails each team member a report that contains only the tasks assigned to him or her. If no tasks are assigned to a particular team member, he or she will not be emailed a report.
- For more details about options that are inherited from Eclipse, see the Eclipse documentation.

XML Files for datagroupConfig and environmentConfig

datagroupConfig XML File Format

```
<tests>
        <test> <!--1 or more-->
                <workspacePath></workspacePath>
                <dataGroups>
                        <dataGroup> <!--1 or more-->
                                <dataGroupName></dataGroupName>
                                <activeDataSourceName></activeDataSourceName>
                        </dataGroup>
                </dataGroups>
        </test>
</tests>
```

environmentConfig XML File Format

```
<tests>
        <test> <!--1 or more-->
                <workspacePath></workspacePath>
                <Environment>
                        <Variable> <!--1 or more-->
                                <Name></Name>
                                <Value></Value>
                        </Variable>
                </Environment>
        </test>
</tests>
```

Note that the `<workspacePath>` element should contain the path to the resource (e.g., .tst) in the workspace—not the path on the file system. It is the **Path** that Eclipse displays if you right-click the .tst file in Eclipse/SOAtest, then choose **Properties**. For example, the path in the following example is **/SOAtest Tutorial/Calculator.tst**.