Configuring Test Configurations

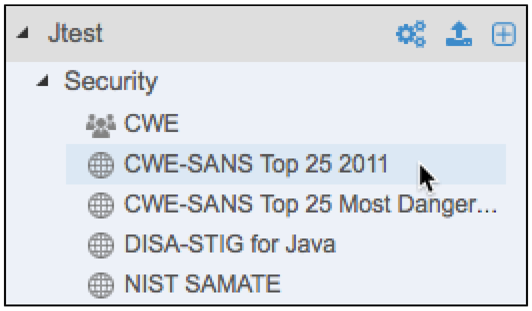
In this chapter:

- Overview
- Scope Tab
- Static Analysis Tab
- Metrics Tab
- Unit Tests Tab
- Static Analysis Settings Tab
- General Tab

Overview

Only users with appropriate permissions can edit test configurations. See About Test Configuration Access.

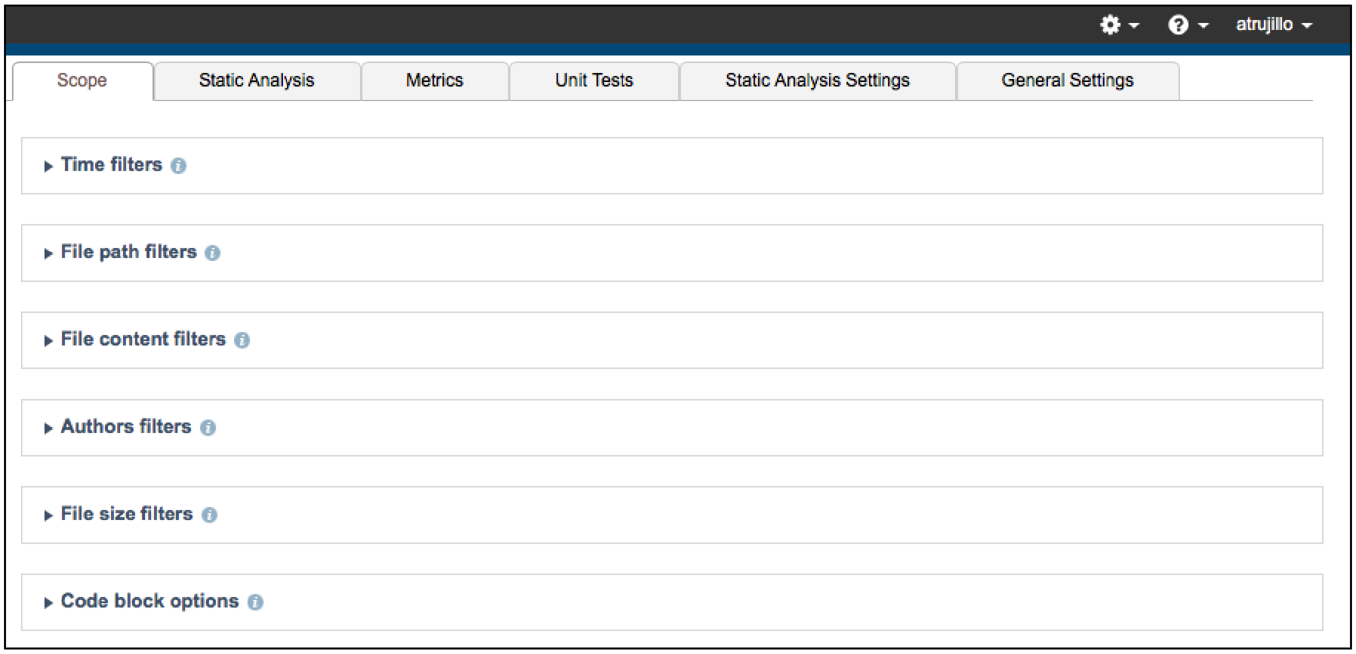
Click on a test configuration in the sidebar to enable editing. Changes affect the test configuration files stored in DTP.



Click on a tab to access a group of related test configuration settings. For additional information about test configuration settings, mouse over an information icon ("i") next to a configuration setting.

Scope Tab

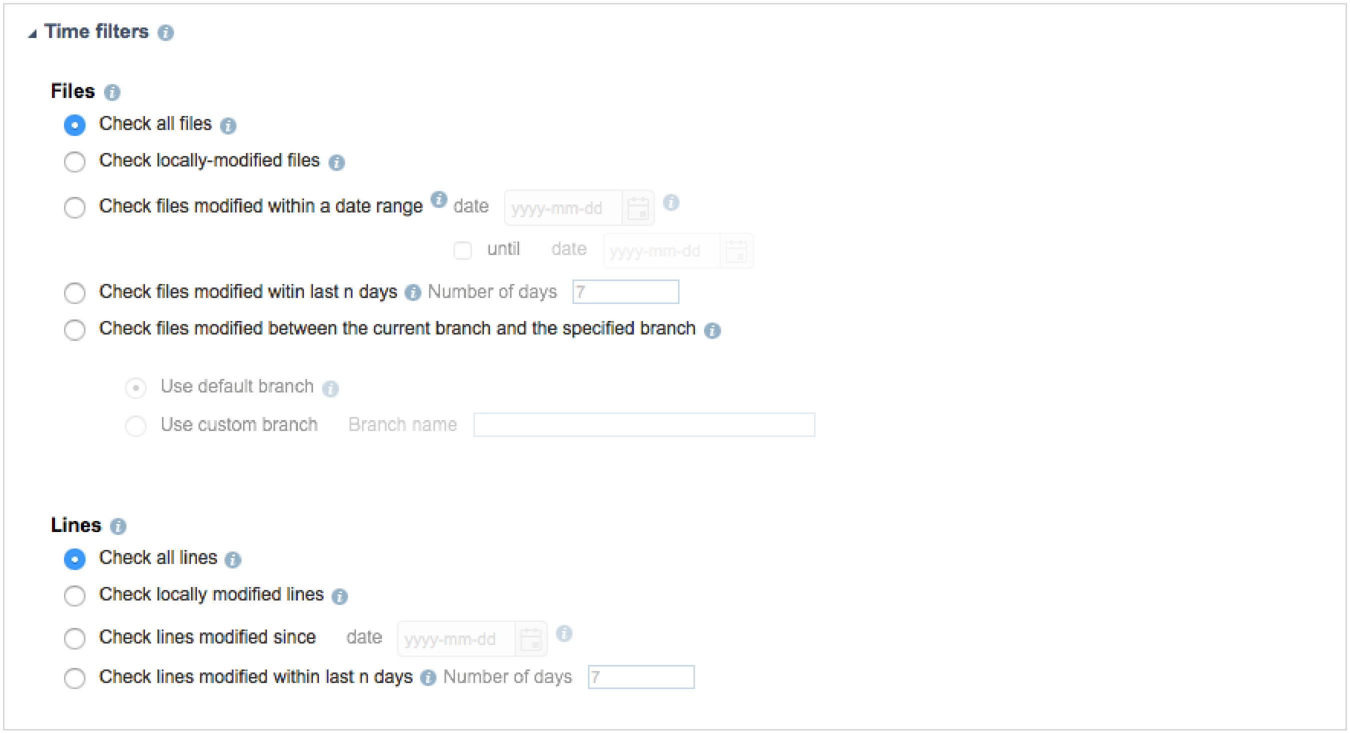
The Scope tab contains a set of filters that you can configure to define the parts of the code that the test configuration should cover. You must connect the code analysis and test execution tool to source control in order to collect scope information. See Configuring Parasoft Test for All Projects. Click **Save** to preserve any changes you make on this tab.



You can configure the following settings.

Time Filters

Expand the Time filters settings to set time-based filters at the file or line level. The time filters enable you to restrict the scope of analysis to a specific date range or period. If the `scope.scontrol` setting is set to `true` and the source control settings for the tool are configured, the modification time is set from the source control history. If `scope.local` is set to `true`, then the modification time is set form the file system of the machine running the tool.


Time filters

Files
[✓ Check all files]
Check locally-modified files
Check files modified within a date range — date: yyyy-mm-dd; until date: yyyy-mm-dd
Check files modified witin last n days — Number of days: 7
Check files modified between the current branch and the specified branch
[✓ Use default branch]
Use custom branch — Branch name:

Lines
[✓ Check all lines]
Check locally modified lines
Check lines modified since — date: yyyy-mm-dd
Check lines modified within last n days — Number of days: 7

You can configure the following settings:

File-level Settings

Check all files	Default. Enable this option to include all files in the scope of the analysis the user has access to.
Check locally-modified files	Enable this option to check only locally-modified files. Files in the source control system will be excluded. The following setting must be configured in the code analysis and test execution tool for this option to take effect: `scope.scontrol=true`
Check files modified within a date range	Enable this option and specify a date range to include in the scope. Files that were modified or added within the specified range will be checked.
Check files modified witin last n days	Enable this option and specify the number of days to include in the scope. Files that were modified or added within the specified number of days will be checked.
Check files modified between the current branch and the specified branch	Enable this option to specify a range of branches to include in the scope. Files that were modified from the user's current branch to the specified branch will be checked. Files that did not change between branches are excluded. You can enable the following options: • Enable the **Use default branch** option to compare the current branch to the branch that the SCM considers default. • Enable the **Use custom branch** and specify a branch to compare with the current branch.

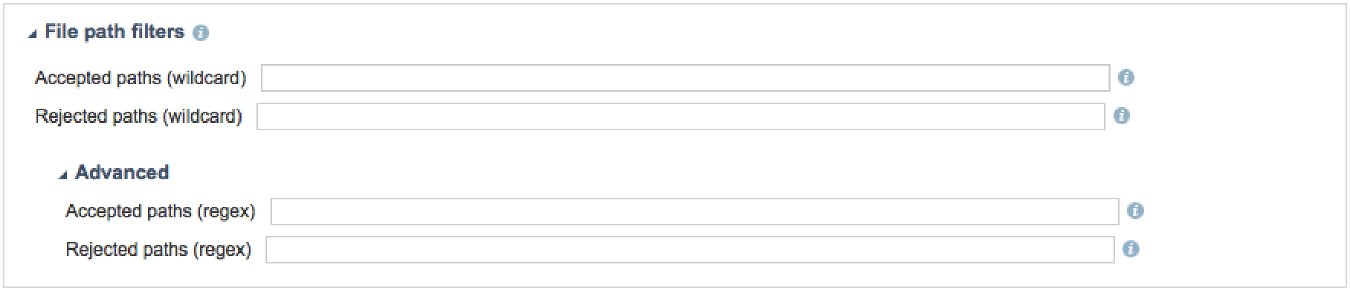
Line-level Settings

Check all lines	Default. Enable this option to include all lines of code in the scope of the analysis the user has access to.
Check locally-modified lines	Enable this option to check only locally-modified lines of code. Lines of code in the source control system will be excluded. The following setting must be configured in the code analysis and test execution tool for this option to take effect: `scope.scontrol=true`

Check lines modified since	Enable this option and specify a cut-off date to include in the scope. Lines of code that were modified or added within the specified range will be checked.
Check lines modified witin last n days	Enable this option and specify the number of days to include in the scope. Lines of code that were modified or added within the specified number of days will be checked.

File Path Filters

Expand the File path filters section to specify file path patterns to include and/or exclude from analysis. Relative paths within a workspace/solution.



The following settings are available:

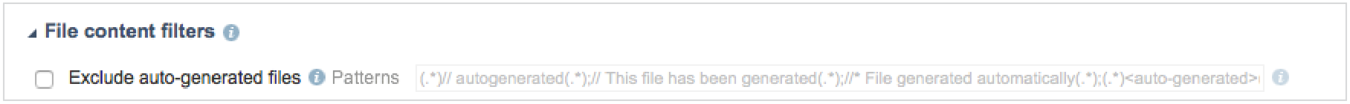
Accepted paths (wildcard)	Specify a comma-separated list of files to include. Wildcards are supported (e.g., *.cpp, *.java, *.cs).
Rejected paths (wildcard)	Specify a comma-separated list of files to exclude. Wildcards are supported (e.g., *.cpp, *.java, *.cs).

Expand the Advanced discloser triangle to use regular expressions to set the file path filters. The following settings are available:

Accepted paths (regex)	Specify a regular expression. Files that match the pattern will be included in the analysis.
Rejected paths (regex)	Specify a regular expression. Files that match the pattern will be excluded in the analysis.

File Content Filters

Expand the File content filters section to specify regular expressions that exclude specific types of files based on content, e.g., auto-generated files.



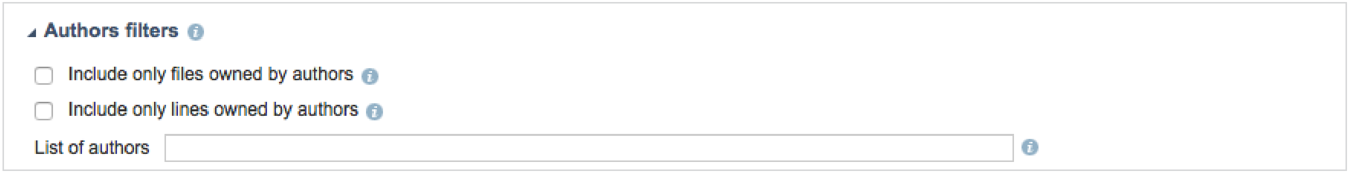
(.*)// autogenerated(.*);// This file has been generated(.*);//* File generated automatically(.*);(.*)<auto-generated>

File filtering takes priority over code block filtering

A potential conflict may occur if you use both filter types at the same time.

Author Filters

Expand the Author filters section to limit the scope of analysis to specific authors. If the `scope.scontrol` setting is set to `true` and the source control settings of the code analysis tool are configured, then file authorship is taken from the source control system. If the `scope.xmlmap` is set to `true` and the XML map settings are configured for the tool, then files authorship is taken from the map.


Authors filters
Include only files owned by authors
Include only lines owned by authors
List of authors

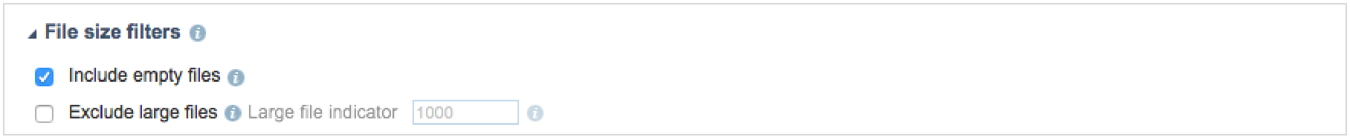
The following options are available:

Include only files owned by authors	Enable this option to only include files owned by the authors specified in the List of authors field.

Include only lines owned by authors	Enable this option to only include lines of code owned by the authors specified in the List of authors field.
List of authors	Specify a comma-separated list of authors whose code should be analyzed.

File Size Filters

Expand the File size filters section to limit the scope of analysis based on file size.

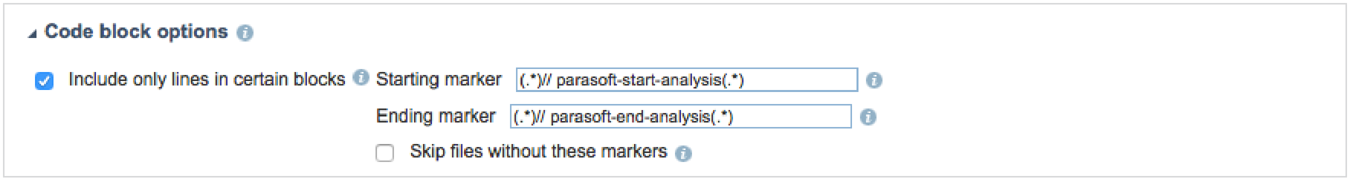


Code Block Options

Expand the Code block options section to define specific blocks of code to include or exclude from the analysis.

File filtering takes priority over code block filtering

A potential conflict may occur if you use both filter types at the same time.


Code block options
Include only lines in certain blocks
Skip files without these markers

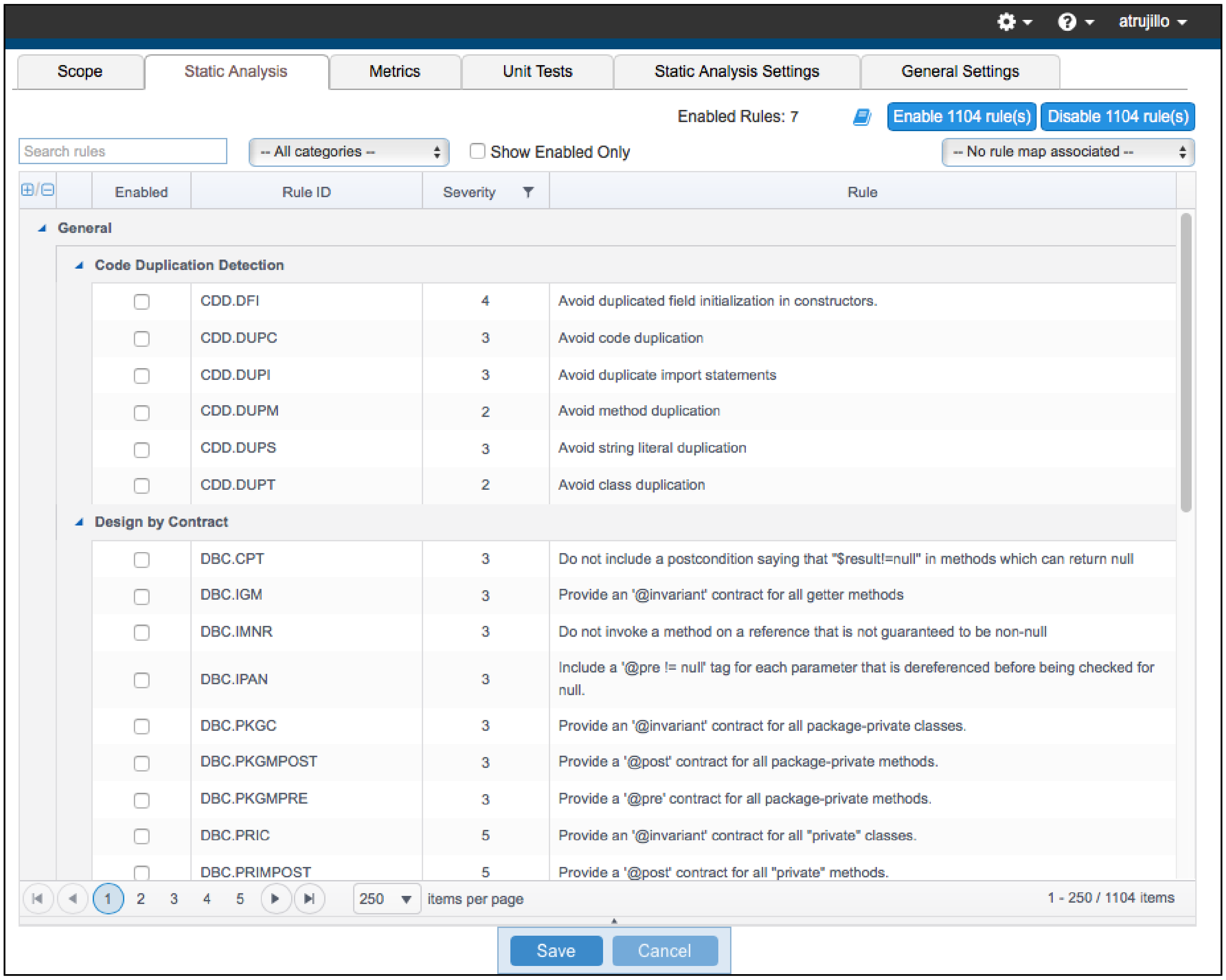
Starting marker: (.*)// parasoft-start-analysis(.*)

Ending marker: (.*)// parasoft-end-analysis(.*)

Include only lines in certain blocks	Enable this option to include only the code defined by the Starting and Ending marker fields in the analysis.
Starting marker	Specify a regular expression to mark the start of the code block that should be analyzed.
Ending marker	Specify a regular expression to mark the beginning of the code block that should be analyzed.
Skip files without these markers	Enable this option skip files that do not include patterns that match the Starting and Ending marker fields.

Static Analysis Tab

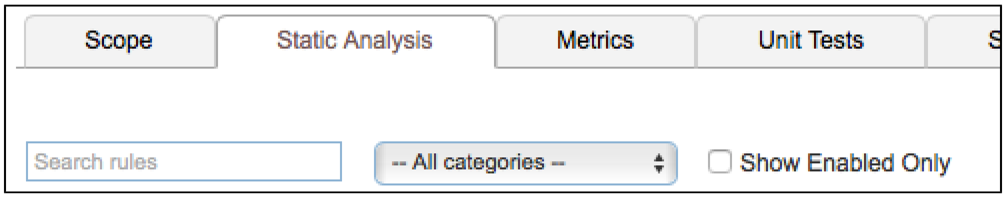
Click the Static Analysis tab to enable/disable the static analysis rules the configuration uses. This page shows all the rules supported by the selected code analysis tool. Click **Save** to preserve any changes you make on this tab.



Enabled	Rule ID	Severity	Rule
General			
Code Duplication Detection			
☐	CDD.DFI	4	Avoid duplicated field initialization in constructors.
☐	CDD.DUPC	3	Avoid code duplication
☐	CDD.DUPI	3	Avoid duplicate import statements
☐	CDD.DUPM	2	Avoid method duplication
☐	CDD.DUPS	3	Avoid string literal duplication
☐	CDD.DUPT	2	Avoid class duplication
Design by Contract			
☐	DBC.CPT	3	Do not include a postcondition saying that "$result!=null" in methods which can return null
☐	DBC.IGM	3	Provide an '@invariant' contract for all getter methods
☐	DBC.IMNR	3	Do not invoke a method on a reference that is not guaranteed to be non-null
☐	DBC.IPAN	3	Include a '@pre != null' tag for each parameter that is dereferenced before being checked for null.
☐	DBC.PKGC	3	Provide an '@invariant' contract for all package-private classes.
☐	DBC.PKGMPOST	3	Provide a '@post' contract for all package-private methods.
☐	DBC.PKGMPRE	3	Provide a '@pre' contract for all package-private methods.
☐	DBC.PRIC	5	Provide an '@invariant' contract for all "private" classes.
☐	DBC.PRIMPOST	5	Provide a '@post' contract for all "private" methods.

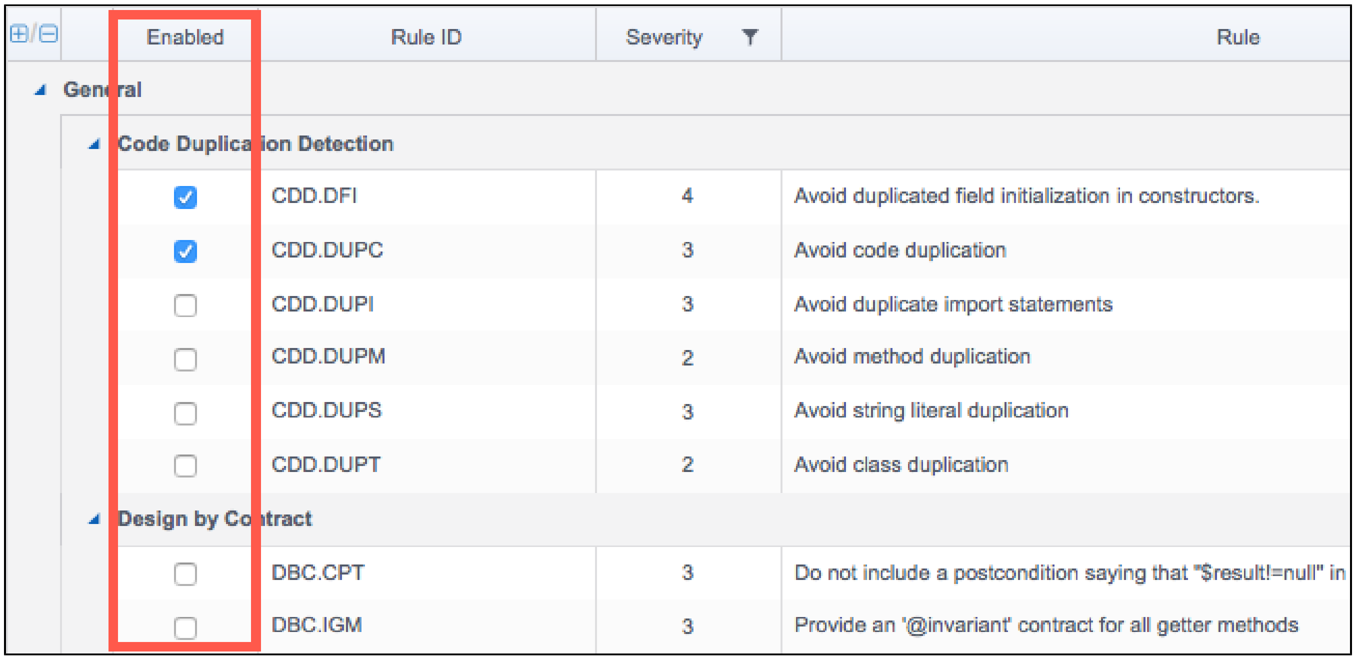
Finding Rules

There are several ways to find specific rules. You can use the search bar to find a specific rule or rule category. You can also use the drop-down menu to filter by category and browse for a rule. Enable the Show Enabled Only option to filter for active rules.



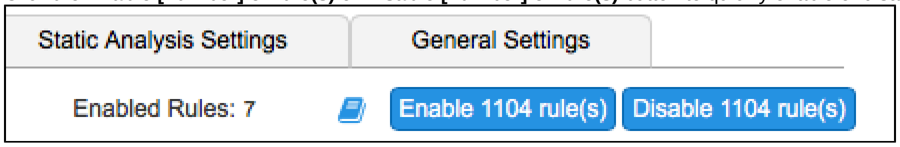
Enabling and Disabling Rules

Rules are grouped by category. Expand a category and enable the rule to use it in the test configuration.



Enabled	Rule ID	Severity	Rule
General			
Code Duplication Detection			
✓	CDD.DFI	4	Avoid duplicated field initialization in constructors.
✓	CDD.DUPC	3	Avoid code duplication
☐	CDD.DUPI	3	Avoid duplicate import statements
☐	CDD.DUPM	2	Avoid method duplication
☐	CDD.DUPS	3	Avoid string literal duplication
☐	CDD.DUPT	2	Avoid class duplication
Design by Contract			
☐	DBC.CPT	3	Do not include a postcondition saying that "$result!=null" in
☐	DBC.IGM	3	Provide an '@invariant' contract for all getter methods

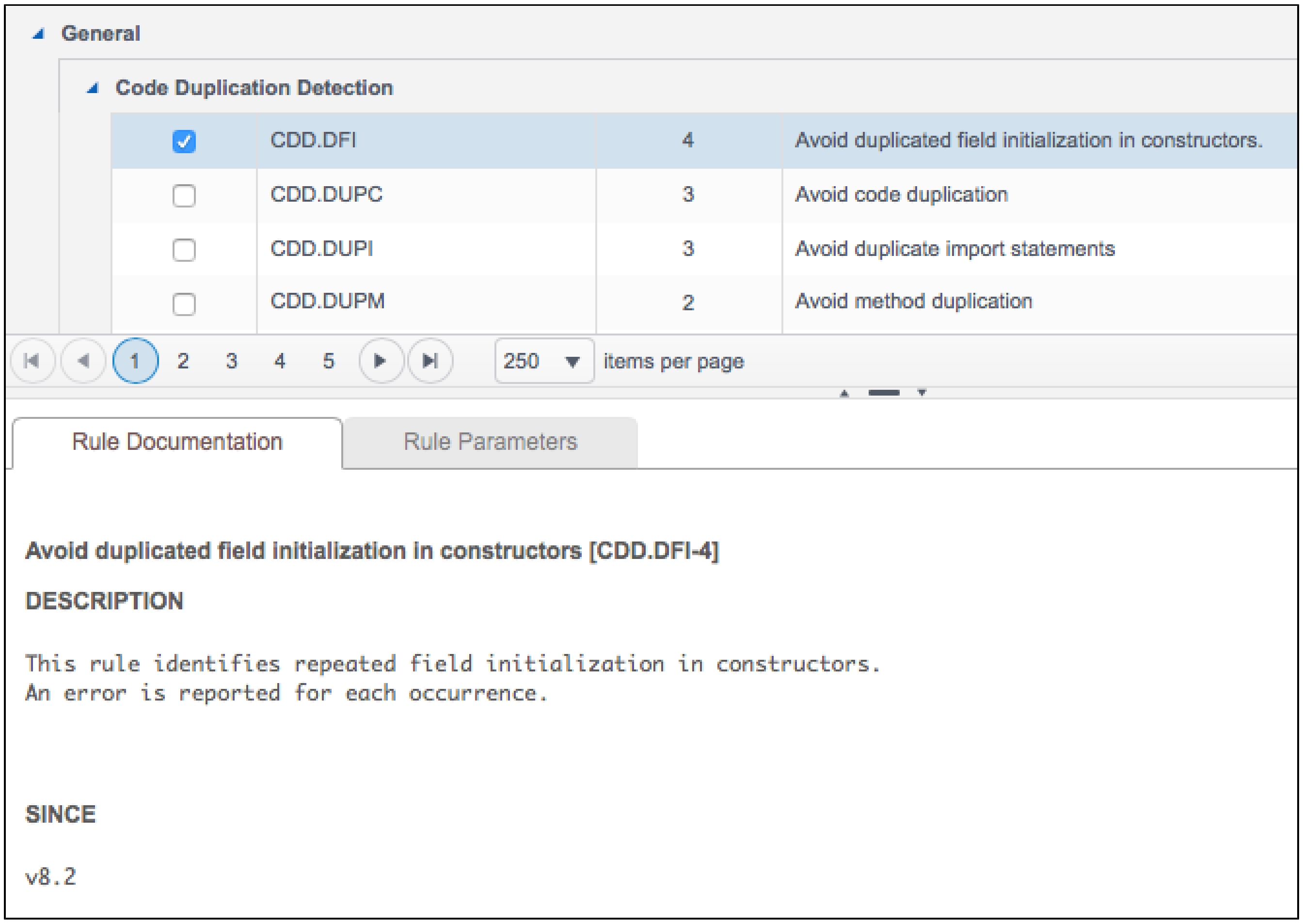
Click the **Enable [number] of rule(s)** or **Disable [number] of rule(s)** button to quickly enable or disable all rules in the configuration.



Enabled Rules: 7

Viewing Rule Documentation

Click on a rule to open the documentation panel.



✓	CDD.DFI	4	Avoid duplicated field initialization in constructors.
☐	CDD.DUPC	3	Avoid code duplication
☐	CDD.DUPI	3	Avoid duplicate import statements
☐	CDD.DUPM	2	Avoid method duplication

Avoid duplicated field initialization in constructors [CDD.DFI-4]

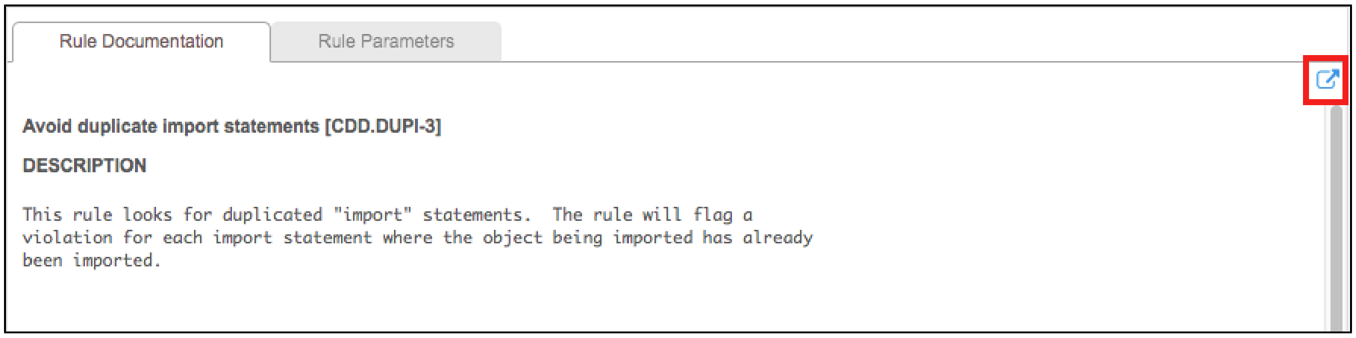
DESCRIPTION

```
This rule identifies repeated field initialization in constructors.
An error is reported for each occurrence.
```

SINCE

```
v8.2
```

You can also open the rule documentation in new browser tab.

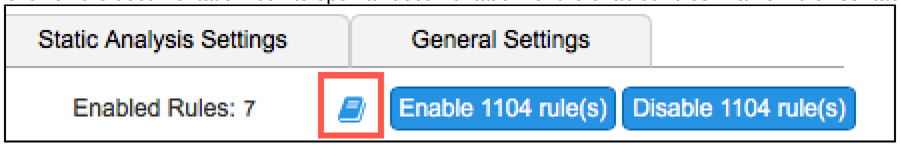


Avoid duplicate import statements [CDD.DUPI-3]

DESCRIPTION

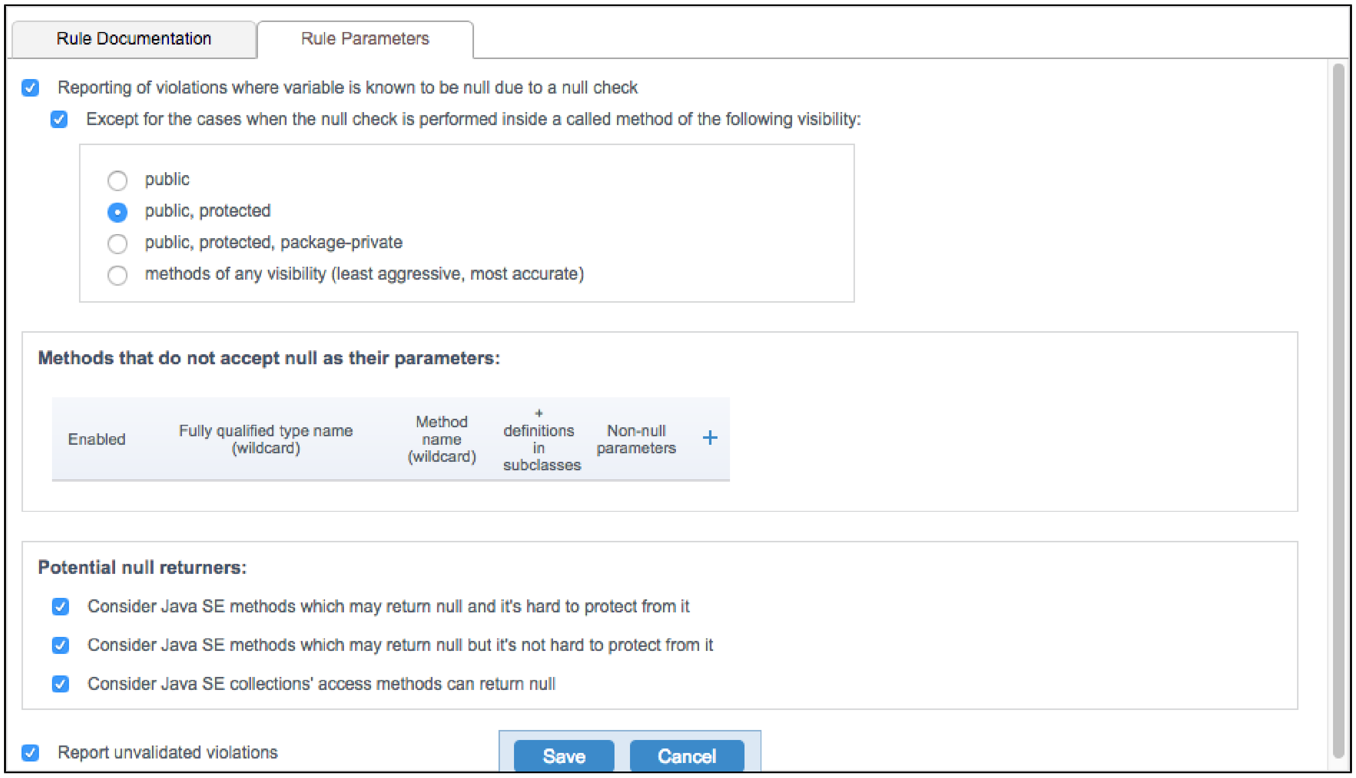
```
This rule looks for duplicated "import" statements.  The rule will flag a
violation for each import statement where the object being imported has already
been imported.
```

Click on the documentation icon to open all documentation for the enabled rules in a new browser tab.



Parameterizing Rules

If the rule can be configured, parameters can be set in the rule options panel. Click on a rule and click the Rule Parameters tab to configure the rule. The options available are specific to each rule.


Rule Documentation | Rule Parameters

- ✓ Reporting of violations where variable is known to be null due to a null check
  - ✓ Except for the cases when the null check is performed inside a called method of the following visibility:
    - ○ public
    - ● public, protected
    - ○ public, protected, package-private
    - ○ methods of any visibility (least aggressive, most accurate)

Methods that do not accept null as their parameters:

Enabled	Fully qualified type name (wildcard)	Method name (wildcard)	+ definitions in subclasses	Non-null parameters	+

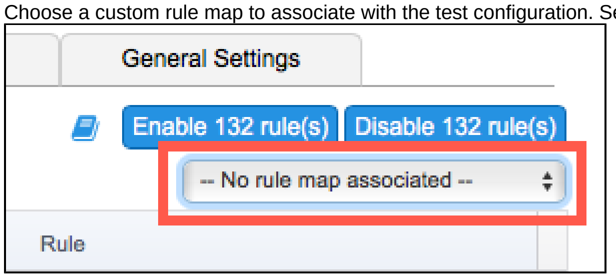
Potential null returners:

- ✓ Consider Java SE methods which may return null and it's hard to protect from it
- ✓ Consider Java SE methods which may return null but it's not hard to protect from it
- ✓ Consider Java SE collections' access methods can return null

✓ Report unvalidated violations

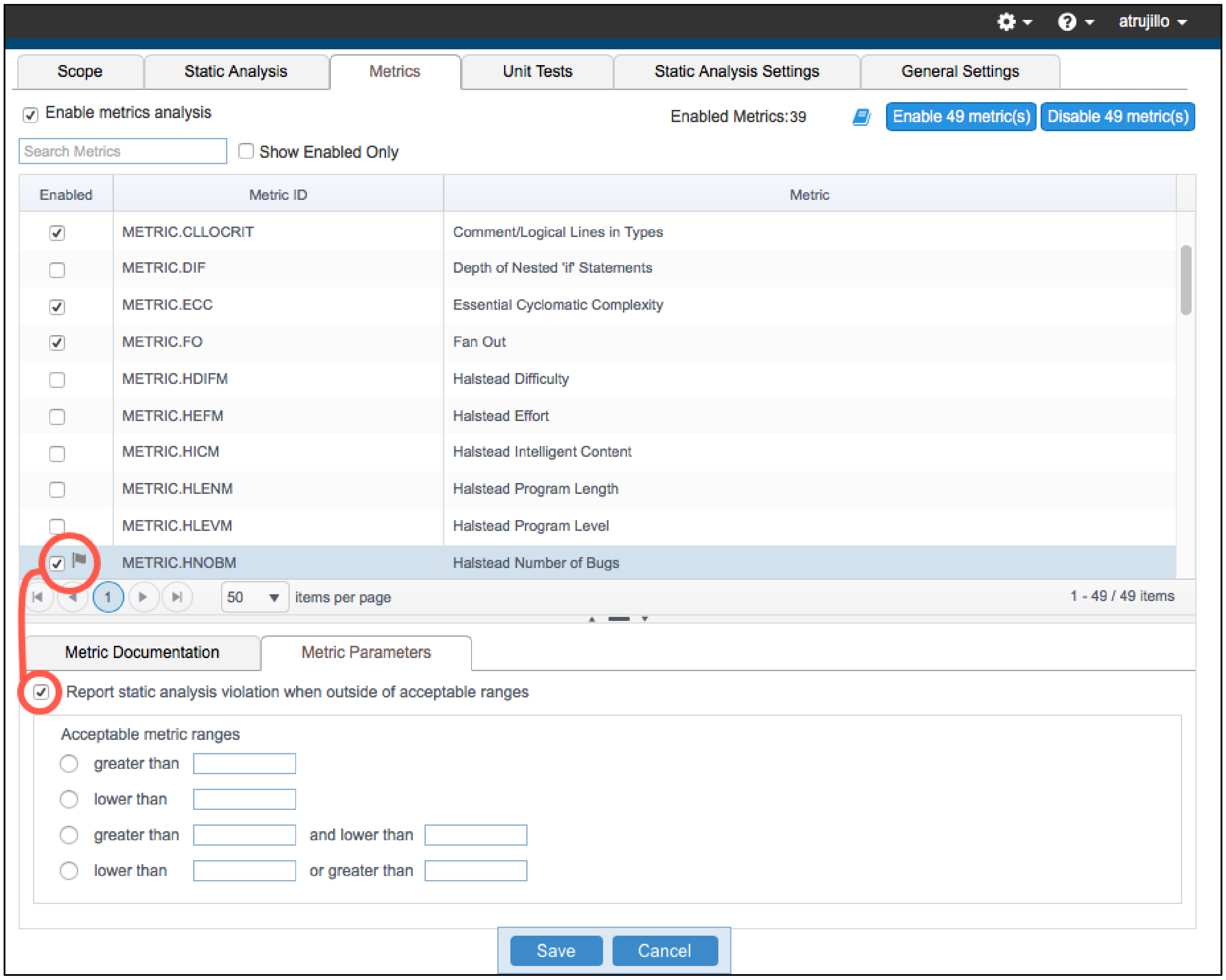
Associating Rule Maps

Choose a custom rule map to associate with the test configuration. See Rule Maps.



Metrics Tab

Click the Metrics tab to enable/disable the metrics collected and calculated during analysis. Click **Save** to preserve any changes you make on this tab.



✓ Enable metrics analysis

Enabled Metrics:39 Enable 49 metric(s) Disable 49 metric(s)

Search Metrics ☐ Show Enabled Only

Enabled	Metric ID	Metric
✓	METRIC.CLLOCRIT	Comment/Logical Lines in Types
☐	METRIC.DIF	Depth of Nested 'if' Statements
✓	METRIC.ECC	Essential Cyclomatic Complexity
✓	METRIC.FO	Fan Out
☐	METRIC.HDIFM	Halstead Difficulty
☐	METRIC.HEFM	Halstead Effort
☐	METRIC.HICM	Halstead Intelligent Content
☐	METRIC.HLENM	Halstead Program Length
☐	METRIC.HLEVM	Halstead Program Level
✓ ⚑	METRIC.HNOBM	Halstead Number of Bugs

1 · 50 items per page · 1 - 49 / 49 items

Metric Documentation | Metric Parameters

✓ Report static analysis violation when outside of acceptable ranges

Acceptable metric ranges

- ○ greater than
- ○ lower than
- ○ greater than and lower than
- ○ lower than or greater than

Save Cancel

You can perform the following actions:

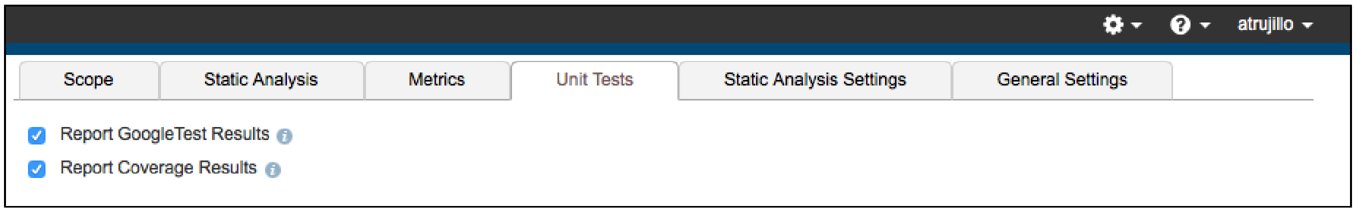
- Enter a metric ID in the search field to locate a specific metric.
- Enable the **Show Enabled Only** option to filter by enabled metrics.
- Click **Enable [n] metric(s)** or **Disable [n] metric(s)** to enable or disable all metrics in the test configuration.
- Enable/disable individual metrics.
- Enable the **Report static analysis violation when outside of acceptable ranges** option to configure an upper and lower threshold for the metric. A flag icon will appear in the Enabled column if this option is enabled.
- Click on a metric and click the **Metric Documentation** tab to view the documentation.

Unit Tests Tab

Click the Unit Tests tab to access controls for unit test execution and coverage data collection.

Test configurations for C++test have the following options:

- **Report Google Test Results**: Enable this option to report Google Test test execution results.
- **Report Coverage Results**: Enable this option to include line coverage monitoring in the results.

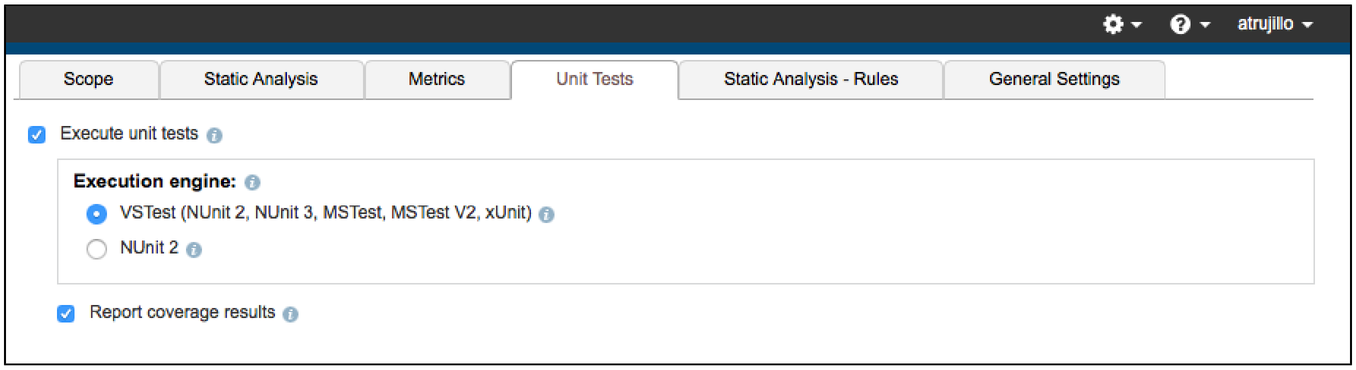


✓ Report GoogleTest Results

✓ Report Coverage Results

Test configurations for dotTEST have the following options:

- **Enable unit tests**: Enable this option to enable unit test execution.
    - **Execution engine**: Choose a unit test execution engine for your testing framework (see the dotTEST documentation for currently-supported frameworks)
- **Report Coverage Results**: Enable this option to include line coverage monitoring in the results.



✓ Execute unit tests

Execution engine:

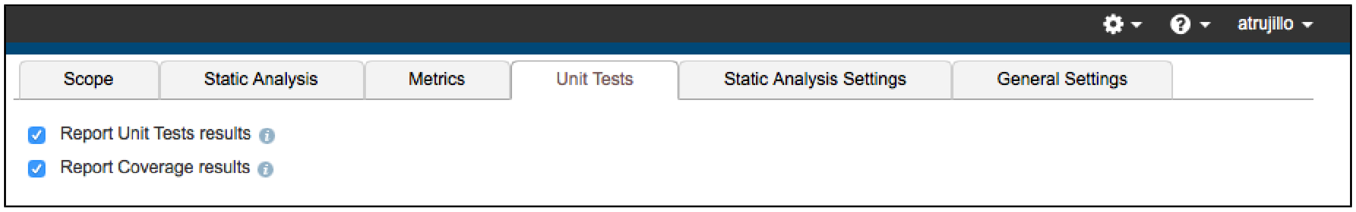
● VSTest (NUnit 2, NUnit 3, MSTest, MSTest V2, xUnit)

○ NUnit 2

✓ Report coverage results

Test configurations for Jtest have the following options:

- **Report Unit Test Results**: Enable this option to report unit test execution results.
- **Report Coverage Results**: Enable this option to include line coverage monitoring in the results.



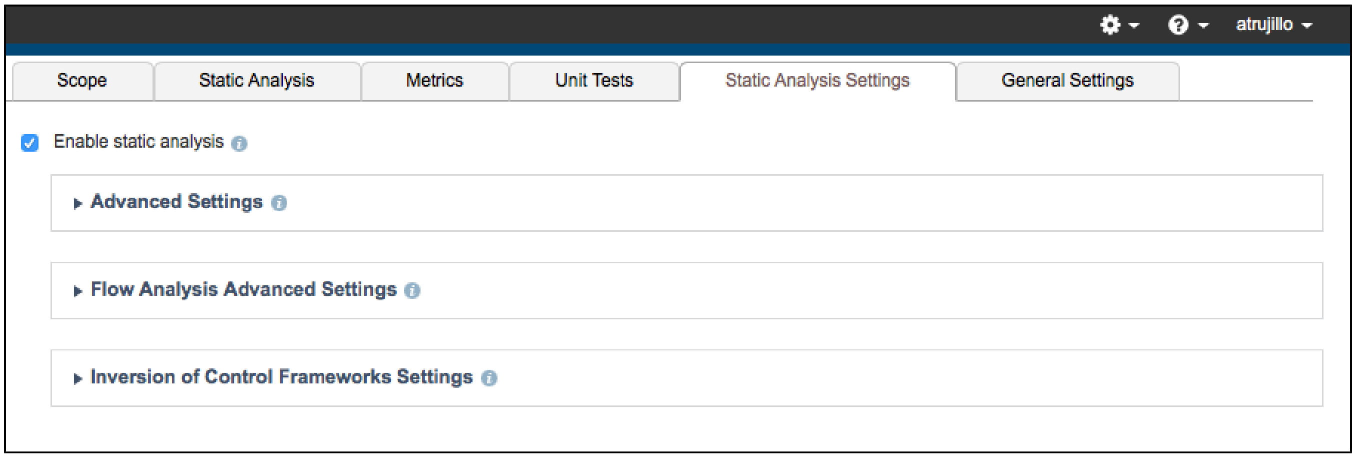
✓ Report Unit Tests results

✓ Report Coverage results

Static Analysis Settings Tab

Click the Static Analysis Settings tab to enable/disable static and flow analysis. The available options may vary depending on the code analysis tool. See the documentation for your tool for additional details.

Click **Save** to preserve any changes you make on this tab.

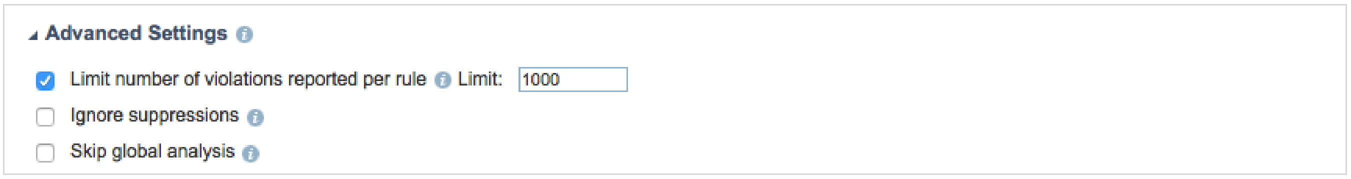


✓ Enable static analysis

Advanced Settings

Expand the Advanced Settings section to enable the following options:

- Set an upper limit on the number of violations that can be reported for each rule.
- Enable/disable suppressions configured on the engine host.
- Enable the **Skip global analysis** option to prevent the collection and use of global data (Jtest only)


Advanced Settings

☑ Limit number of violations reported per rule Limit: 1000

☐ Ignore suppressions

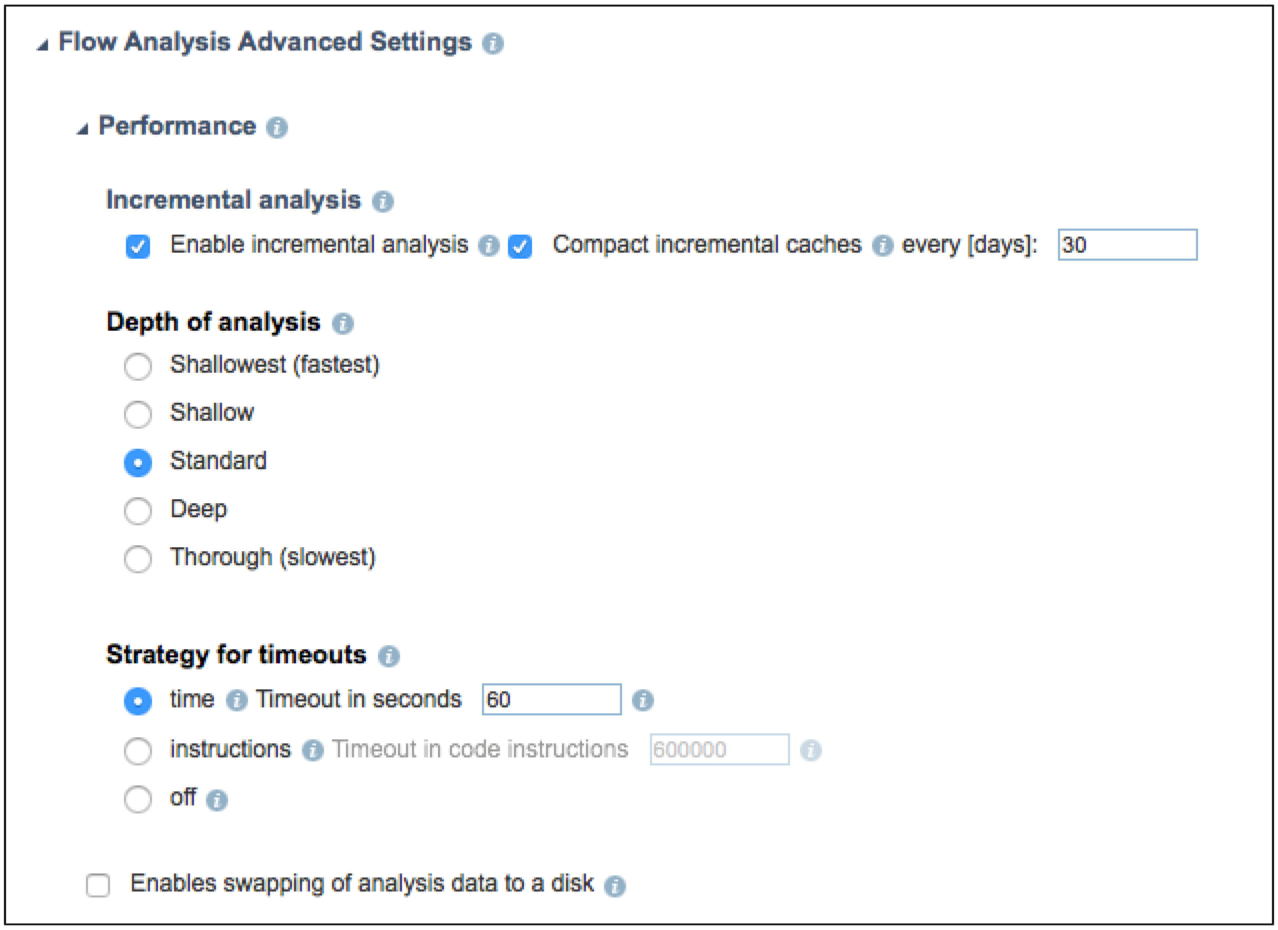
☐ Skip global analysis

Flow Analysis Advanced Settings

Expand the Flow Analysis Advanced Settings section to configure settings related to performance, reporting verbosity, null-checking method parameterization, and resources checked.

Performance Settings

Expand this section to access options related to depth of analysis and machine performance. By default, flow analysis performs a complete analysis of the scope, which can take considerable time when running on large code bases.


Flow Analysis Advanced Settings

Performance

Incremental analysis

☑ Enable incremental analysis ☑ Compact incremental caches every [days]: 30

Depth of analysis

○ Shallowest (fastest)

○ Shallow

● Standard

○ Deep

○ Thorough (slowest)

Strategy for timeouts

● time Timeout in seconds 60

○ instructions Timeout in code instructions 600000

○ off

☐ Enables swapping of analysis data to a disk

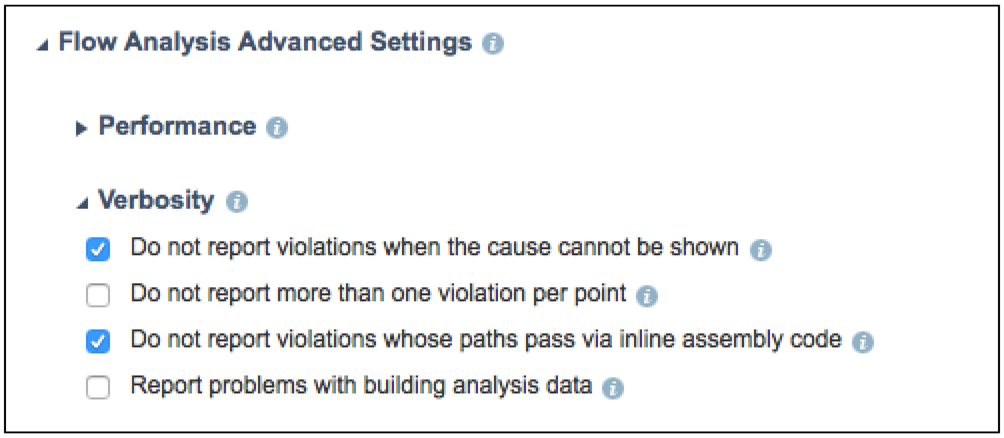
You can configure the following options:

Incremental analysis	Enable the **Enable incremental analysis** option to run flow analysis in incremental mode. Incremental mode reduces the time required to run analysis during nightly runs on a single code base that changes from day to day. Enable the **Compact incremental caches** option and specify how frequently unnecessary data that may have been cached is removed. *This option is available for C++test only.*
Depth of analysis	Choose the depth of flow analysis. A deeper setting results in more findings, but may results in slower analysis and greater memory consumption.

Strategy for timeouts	Configure the timeout settings for flow analysis. Enable the **time** option and specify how many seconds flow analysis should wait before timing out when analyzing a single point in the code. Enable the **instructions** option and specify the maximum number of flow analysis instructions allowed when analyzing a single point in the code. Enable **off** to disable timeouts.
Enables swapping of analysis data to disk	Enable this option to allow the flow analysis tool to write the data necessary for analysis to disk. This prevents flow analysis from consuming all available memory when analyzing a large project in which the data represents a semantic model of the analyzed source code.

Verbosity Settings

Expand this section to access options related to how much detail is included in the report.



Flow Analysis Advanced Settings

Performance

Verbosity

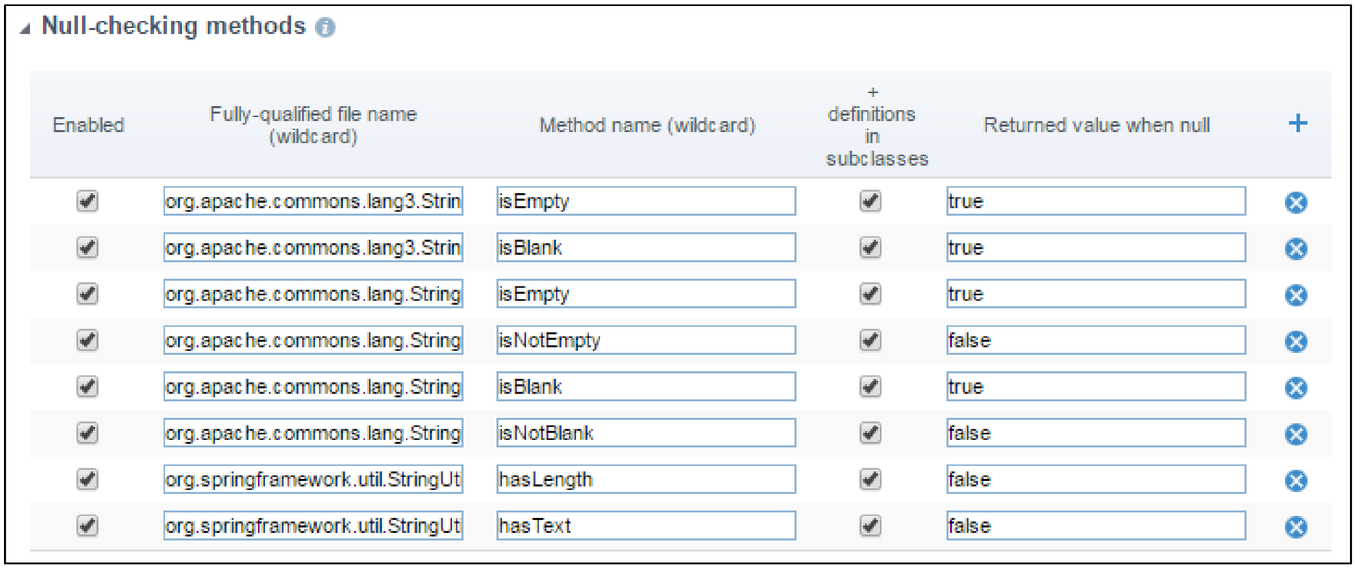
- [✓] Do not report violations when the cause cannot be shown
- [ ] Do not report more than one violation per point
- [✓] Do not report violations whose paths pass via inline assembly code
- [ ] Report problems with building analysis data

The following options are available:

Do not report violations when cause cannot be shown	Enable this option if you want to exclude violations that lack a cause from being reported. Complete paths from the cause to the point of violation are reported if this option is enabled, but violation causes are not reported for paths that have multiple causes.
Do not report more than one violation per point	Enable this option to restrict the number reported violations when the cause or point of violation is shared.
Do Not report violations whose paths pass via inline assembly code	Enable this option to report violations whose paths are passed with inline assembly code instructions. *C/C++test only*.
Report problems with building analysis data	Enable this option to report a set up problem if flow analysis encounters a problem while building analysis data.

Null-checking Methods Settings

Expand the Null-checking methods section to specify the expected return value when a null parameter is passed to a method. This reduces false positives and excessive paths that would normally be built when the return value of a null variable is unknown. **These settings are not available for C/C++test**.

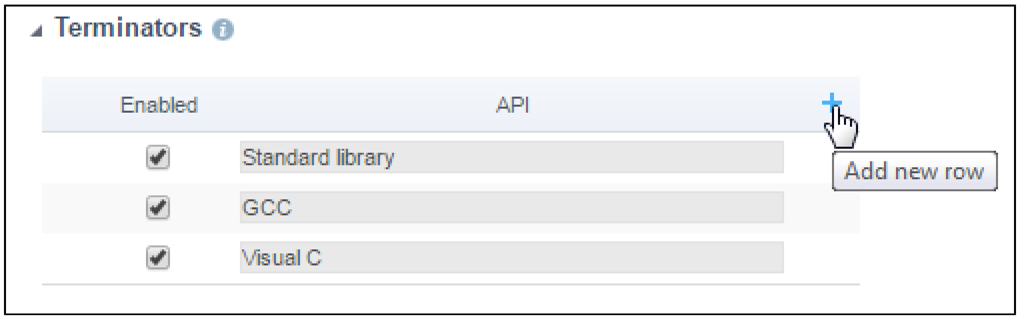

Null-checking methods

Enabled	Fully-qualified file name (wildcard)	Method name (wildcard)	+ definitions in subclasses	Returned value when null	+
✓	org.apache.commons.lang3.Strin	isEmpty	✓	true	✖
✓	org.apache.commons.lang3.Strin	isBlank	✓	true	✖
✓	org.apache.commons.lang.String	isEmpty	✓	true	✖
✓	org.apache.commons.lang.String	isNotEmpty	✓	false	✖
✓	org.apache.commons.lang.String	isBlank	✓	true	✖
✓	org.apache.commons.lang.String	isNotBlank	✓	false	✖
✓	org.springframework.util.StringUt	hasLength	✓	false	✖
✓	org.springframework.util.StringUt	hasText	✓	false	✖

Click the add button (**+**) to add a new parameterization. Click the delete button (**x**) to remove parameterizations.

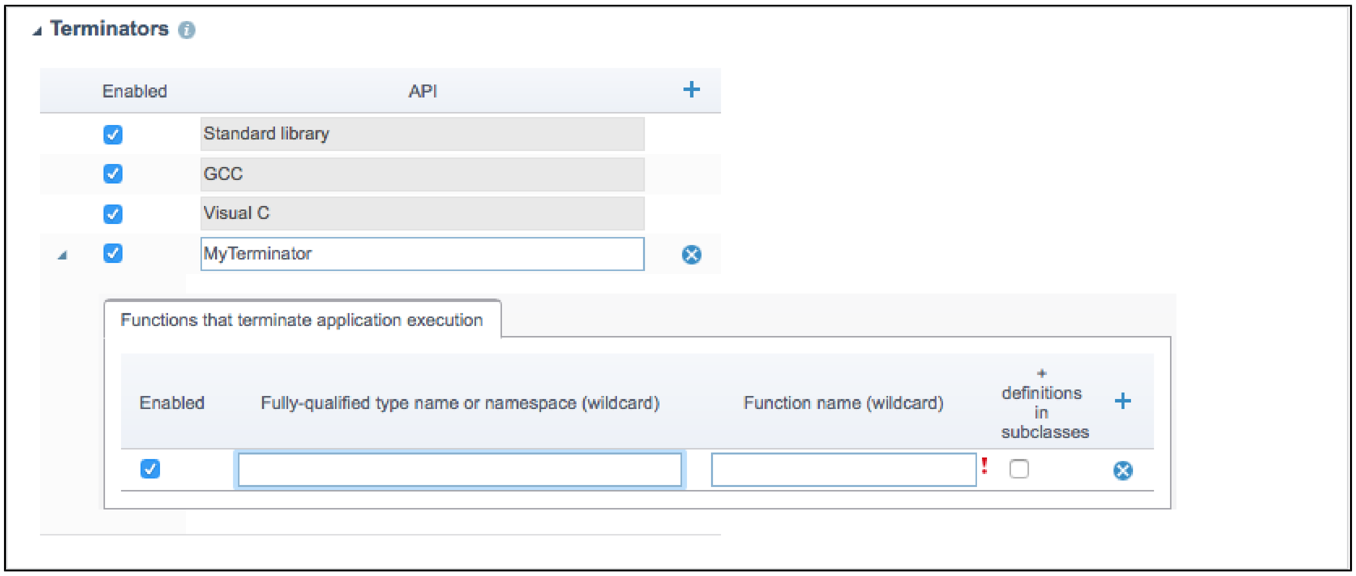
Terminators Settings

Expand the Terminators section to define functions that terminate application execution. **These settings are available for C/C++test only**.


Terminators

Enabled	API	+
✓	Standard library	
✓	GCC	
✓	Visual C	

Click the add button (**+**) to add a new terminating function API.


Terminators

Enabled	API	+
✓	Standard library	
✓	GCC	
✓	Visual C	
✓	MyTerminator	✖

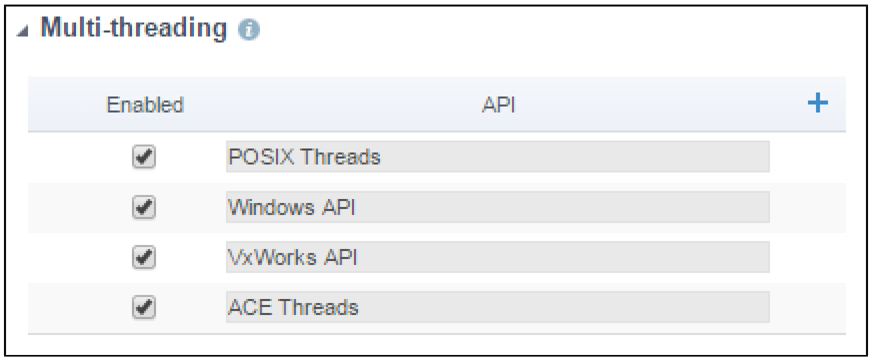
Functions that terminate application execution

Enabled	Fully-qualified type name or namespace (wildcard)	Function name (wildcard)	+ definitions in subclasses	+
✓		!	☐	✖

Configure the terminator in the fields provided. See the C/C++test documentation for details.

Multi-threading Settings

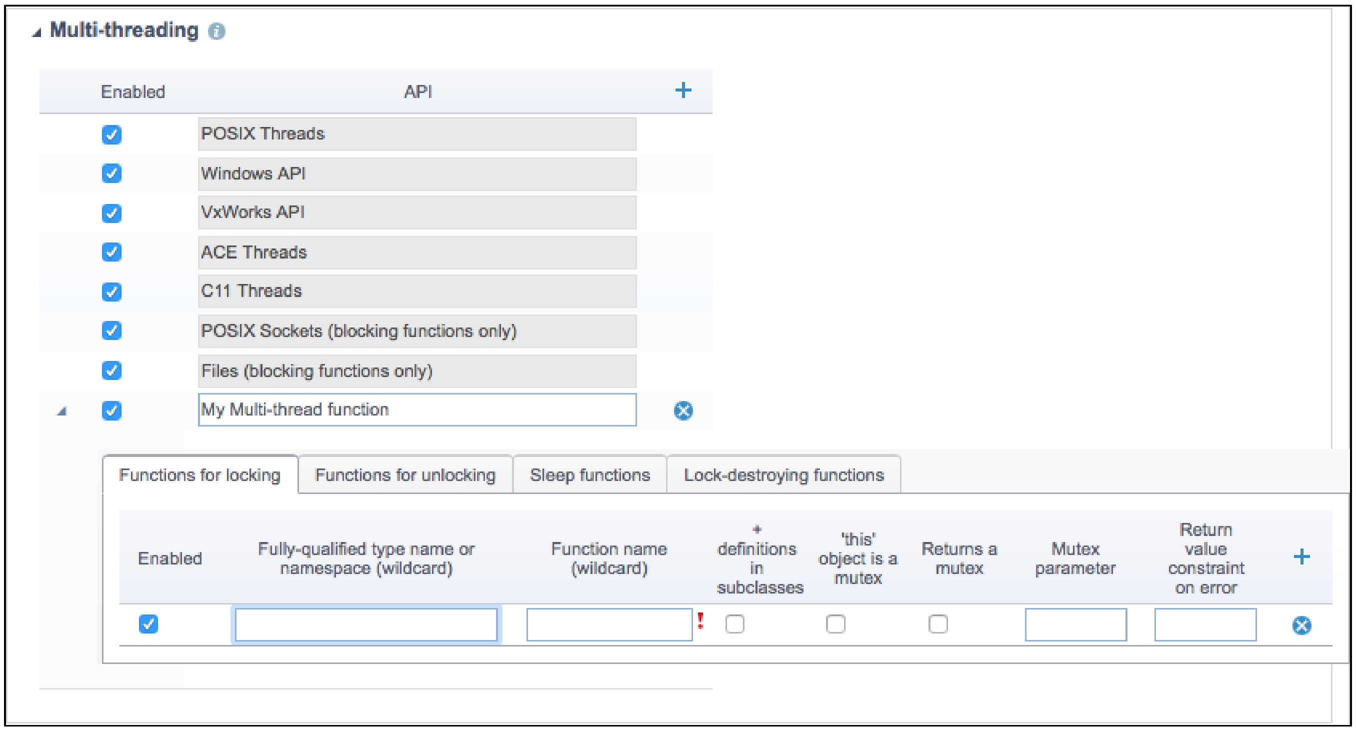
Expand the Multi-threading section to define functions for synchronization between threads and to activate/deactivate the multi-threading functions already listed. The options specified here affect the rules from the BD.TRS (Threads and Synchronization) category. **These settings are available for C/C++test only.**



Multi-threading

Enabled	API
✓	POSIX Threads
✓	Windows API
✓	VxWorks API
✓	ACE Threads

Click the add button (**+**) to add a new function.



Multi-threading

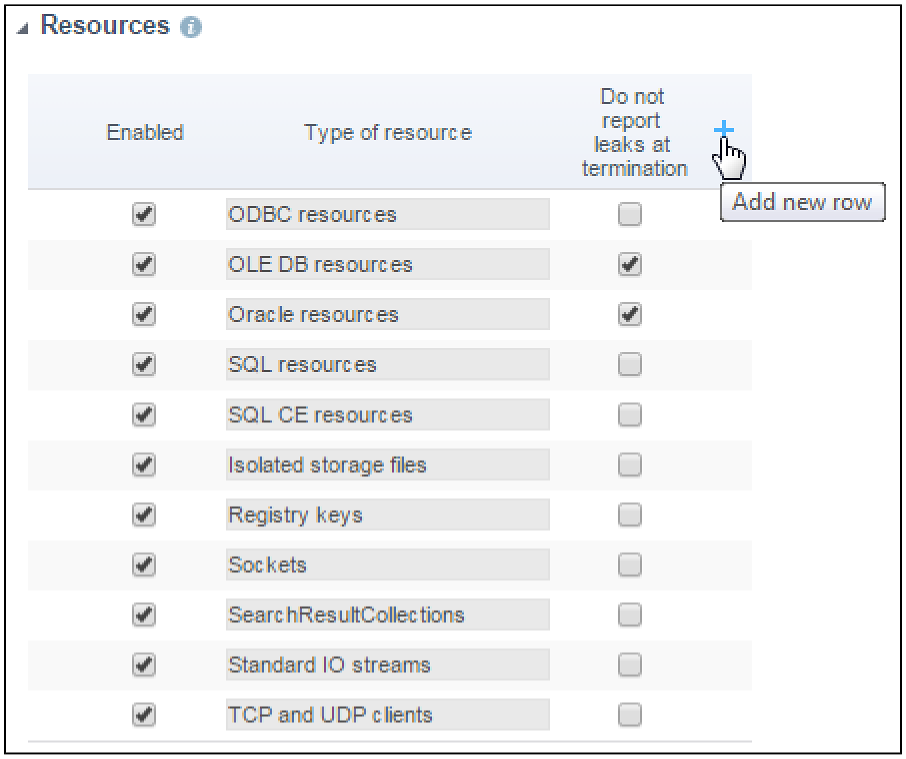
Enabled	API
✓	POSIX Threads
✓	Windows API
✓	VxWorks API
✓	ACE Threads
✓	C11 Threads
✓	POSIX Sockets (blocking functions only)
✓	Files (blocking functions only)
✓	My Multi-thread function

Enabled	Fully-qualified type name or namespace (wildcard)	Function name (wildcard)	+ definitions in subclasses	'this' object is a mutex	Returns a mutex	Mutex parameter	Return value constraint on error
✓		!	☐	☐	☐		

Specify a name and configure the function. See the C/C++test documentation for details.

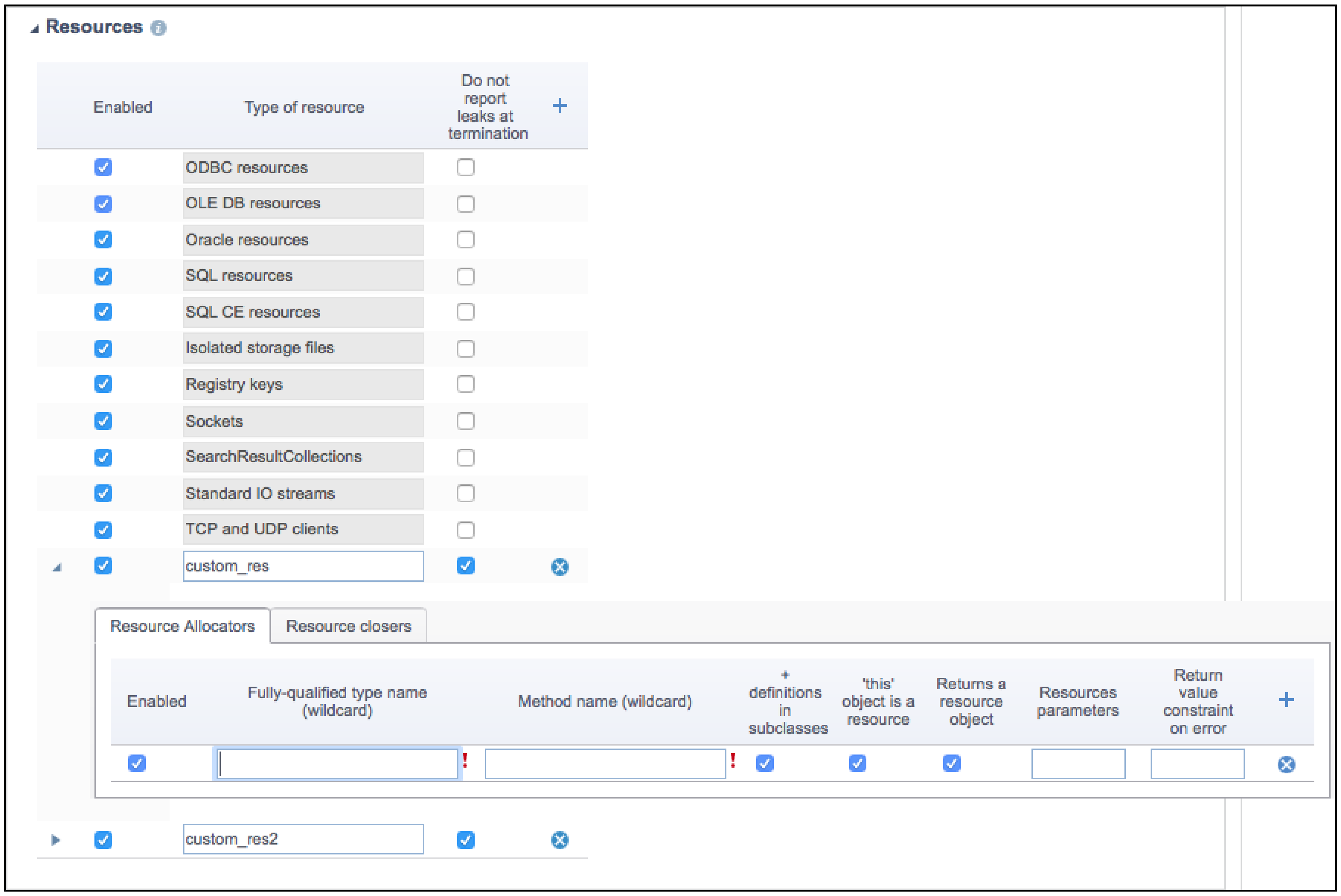
Resources Settings

These settings allow you to define which resources should be checked by the resources rules (BD.RES category). These rules verify the correct usage of all resources that are defined and enabled in this tab.


Resources

Enabled	Type of resource	Do not report leaks at termination
✓	ODBC resources	☐
✓	OLE DB resources	✓
✓	Oracle resources	✓
✓	SQL resources	☐
✓	SQL CE resources	☐
✓	Isolated storage files	☐
✓	Registry keys	☐
✓	Sockets	☐
✓	SearchResultCollections	☐
✓	Standard IO streams	☐
✓	TCP and UDP clients	☐

Click the add button (**+**) to add a new resource.


Resources

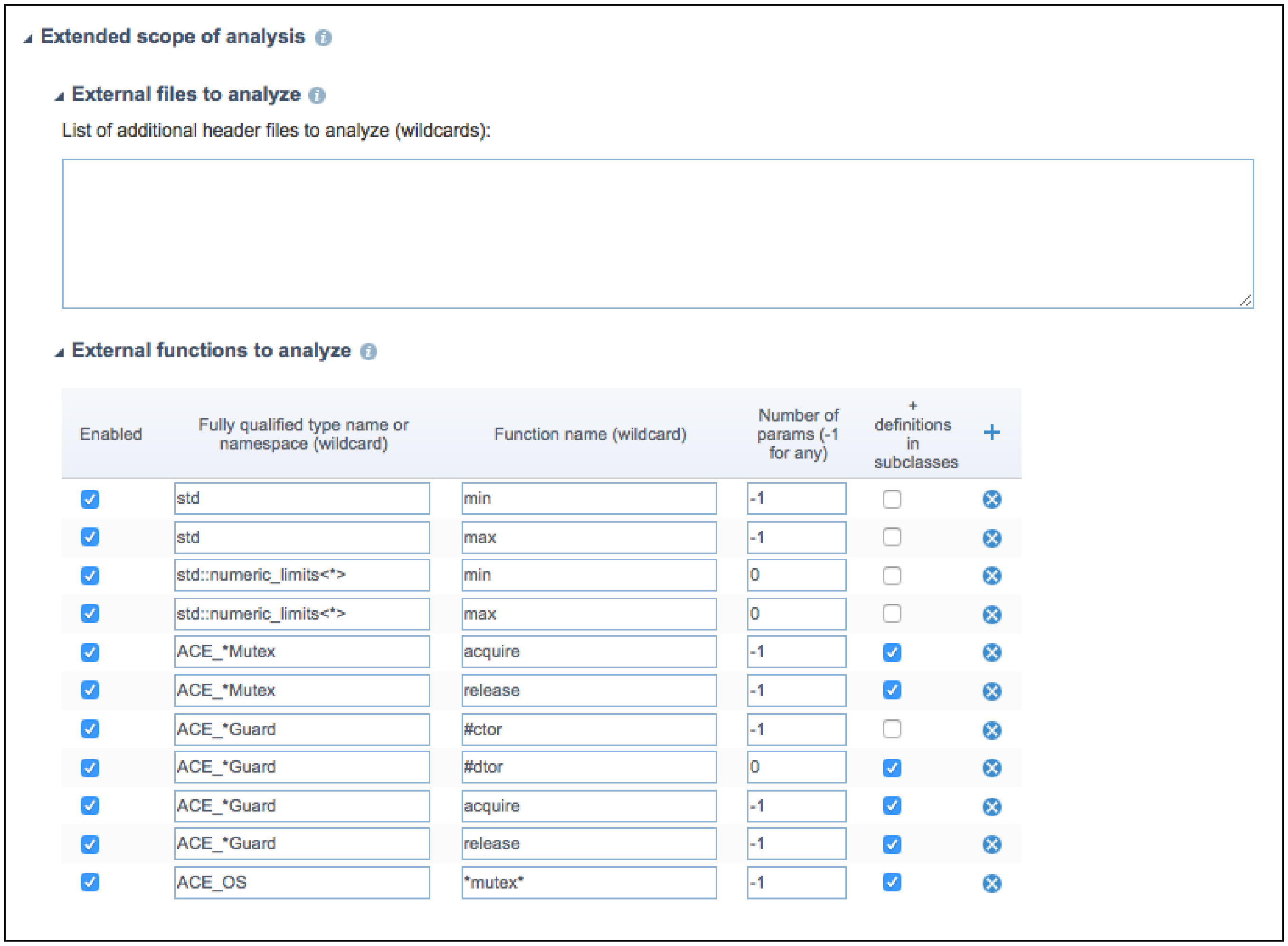
Enabled	Type of resource	Do not report leaks at termination
✓	ODBC resources	☐
✓	OLE DB resources	☐
✓	Oracle resources	☐
✓	SQL resources	☐
✓	SQL CE resources	☐
✓	Isolated storage files	☐
✓	Registry keys	☐
✓	Sockets	☐
✓	SearchResultCollections	☐
✓	Standard IO streams	☐
✓	TCP and UDP clients	☐
✓	custom_res	✓
✓	custom_res2	✓

Enabled	Fully-qualified type name (wildcard)	Method name (wildcard)	+ definitions in subclasses	'this' object is a resource	Returns a resource object	Resources parameters	Return value constraint on error
✓	!	!	✓	✓	✓		

Specify a name and configure the resource.

Extended scope of analysis Settings

When performing code analysis, Flow Analysis processes definitions of functions that are defined in source and header files under test. Functions that are defined in header files outside the testing scope are not analyzed, and Flow Analysis is not aware of their semantics. If Flow Analysis requires information about function definitions that are defined in header files outside the testing scope. **These settings are available for C/C++test only.**



Extended scope of analysis

External files to analyze

List of additional header files to analyze (wildcards):

External functions to analyze

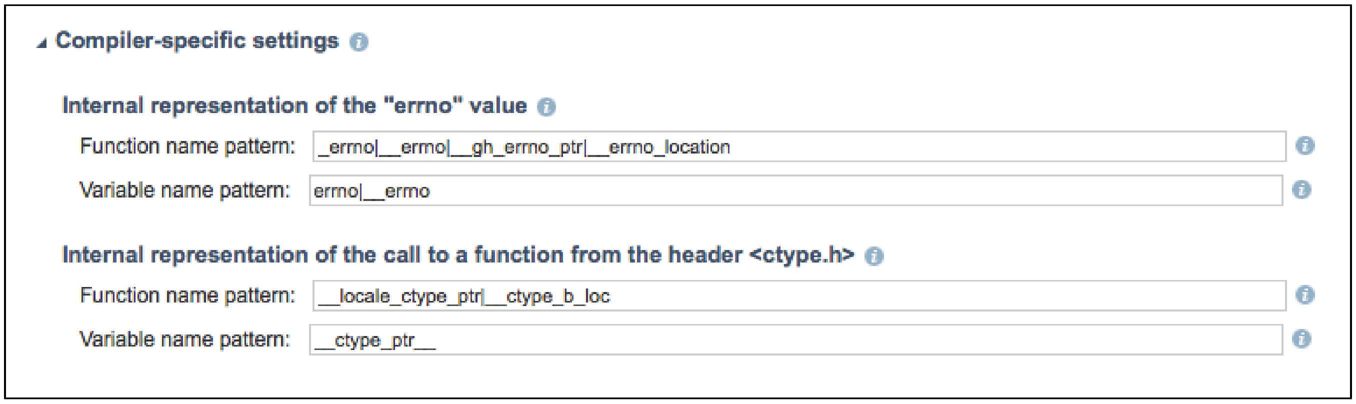
Enabled	Fully qualified type name or namespace (wildcard)	Function name (wildcard)	Number of params (-1 for any)	+ definitions in subclasses
✓	std	min	-1	☐
✓	std	max	-1	☐
✓	std::numeric_limits<*>	min	0	☐
✓	std::numeric_limits<*>	max	0	☐
✓	ACE_*Mutex	acquire	-1	✓
✓	ACE_*Mutex	release	-1	✓
✓	ACE_*Guard	#ctor	-1	☐
✓	ACE_*Guard	#dtor	0	✓
✓	ACE_*Guard	acquire	-1	✓
✓	ACE_*Guard	release	-1	✓
✓	ACE_OS	*mutex*	-1	✓

You can configure the following settings:

External files to analyze	Specifies absolute paths to additional header files to be analyzed by Flow Analysis. Use wildcards to specify the pattern.
External functions to analyze	Specifies additional functions to be analyzed by Flow Analysis. Complete the table with the following information: • **Enabled**: specifies whether the function should be considered during analysis • **Fully-qualified type name or namespace (wildcard)**: the fully-qualified name of the type or namespace where the function is declared. Use '*' if you want to describe a function declared in any type or namespace, or a global function declared outside of any type. • **Function name (wildcard)**: the name of the function. '*' can be used to denote any number of any symbols. • **Number of parameters**: specifies the number of function's parameters. '-1' can be used to denote any number of parameters. • **+ definitions in subclasses**: a checkbox that indicates whether the definitions (of functions with the given name) in subclasses should be included as well. Note that this applies to both instance and static functions.

Compiler-specific Settings

These settings allow you to define advanced compiler-specific arguments. **These settings are available for C/C++test only.**


Compiler-specific settings

Internal representation of the "errno" value

Function name pattern: _errno|__errno|__gh_errno_ptr|__errno_location

Variable name pattern: errno|__errno

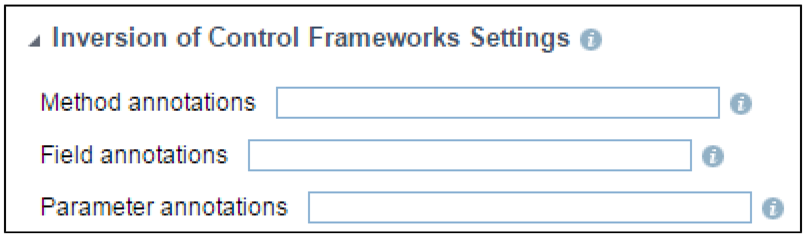
Internal representation of the call to a function from the header <ctype.h>

Function name pattern: __locale_ctype_ptr|__ctype_b_loc

Variable name pattern: __ctype_ptr__

Inversion of Control Framework Settings

Expand the Inversion of Control Framework Settings section to specify annotations used for injecting/initialization values during runtime. Enter a comma-separated list of fully qualified names to use as annotations. **These settings are available for Jtest only.**


Inversion of Control Frameworks Settings

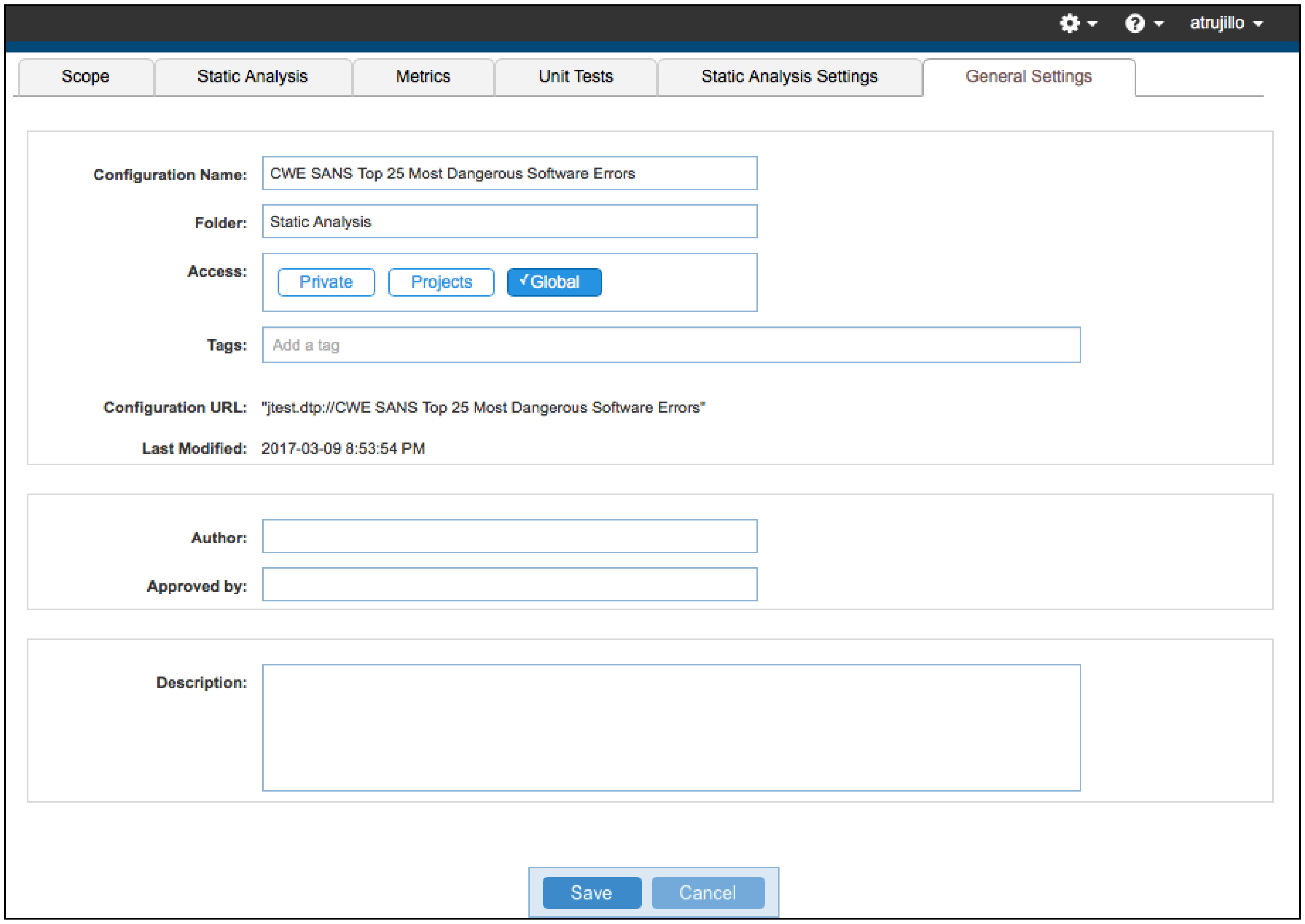
Method annotations

Field annotations

Parameter annotations

General Tab

Click on the **General** tab to view and edit metadata associated with the configuration. Click **Save** to preserve any changes you make on this tab.



Configuration Name: CWE SANS Top 25 Most Dangerous Software Errors

Folder: Static Analysis

Access: Private | Projects | [✓ Global]

Tags: Add a tag

Configuration URL: "jtest.dtp://CWE SANS Top 25 Most Dangerous Software Errors"

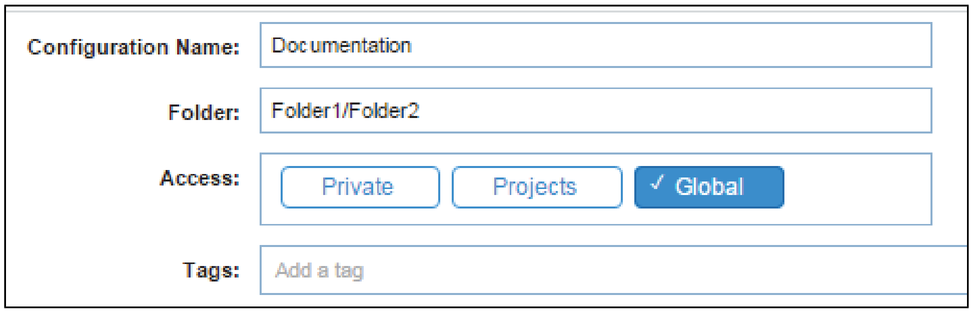
Last Modified: 2017-03-09 8:53:54 PM

Author:

Approved by:

Description:

The following settings and information are available:

Configuration Name	You can rename the test configuration.
Folder	Enter a name in the Folder field to change the location of the test configuration. Entering the name of an existing folder moves the test configuration to that location. If the name you specify doesn't exist, a new folder will be created and the test configuration moved into it. You can also nest folders by placing a forward slash (/) between folder names.  Configuration Name: Documentation Folder: Folder1/Folder2 Access: Private | Projects | [✓ Global] Tags: Add a tag
Access	Choose test configuration accessibility. See About Test Configuration Access.
Tags	Add or remove tags, which searchable terms attached to the configuration.
Configuration URL	View the configuration file URL that is consumed by the code analysis and test execution tool. This cannot be changed.
Last Modified	View when the test configuration was added or updated.

Author	Add an author. The author may be resource for team members who can provide guidance.
Approver	Add an approver. The approver may be someone who can provide additional information about the test configuration or serve as a quality gate.
Description	Add a description to provide additional information about the configuration. Descriptions may help team members understand the goals associated with the test configuration.