Metrics Calculation

This topic explains how to use C++test to calculate metrics. Sections include:

About Metrics

C++test 's metrics analysis calculates various code metrics, such as code complexity, coupling between objects, and lack of cohesion, to help you assess your code base and monitor changes. More specifically, calculating and tracking metrics helps you to:

- Better understand code complexity and what other classes a change in the code may affect. This helps you make more informed decisions as to how to modify, refactor, and test it.
- Identify some of the symptoms of poor design and understand potential weak points in the code.

Run the **Metrics** built-in test configuration during analysis to execute metrics analysis rules.

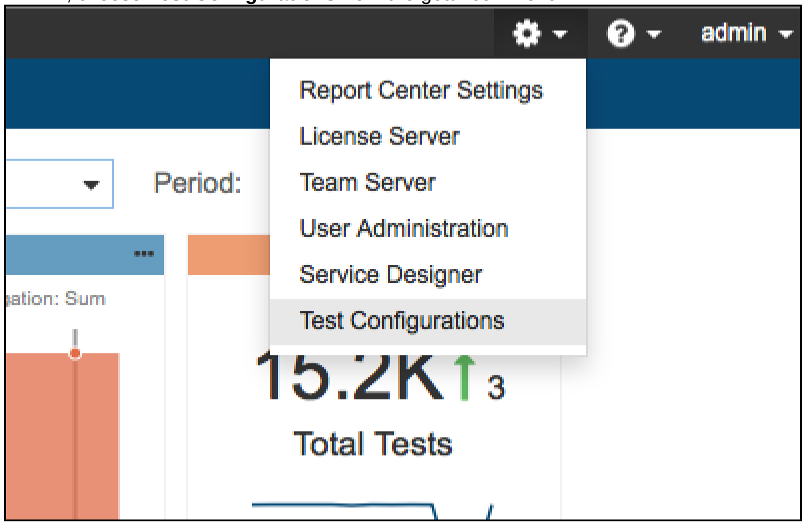
Reviewing Metrics Results

Metrics analysis results are available on DTP server; see Connecting to DTP for information how to connect with DTP. Metrics results are not included in local HTML or XML reports generated by C++test.

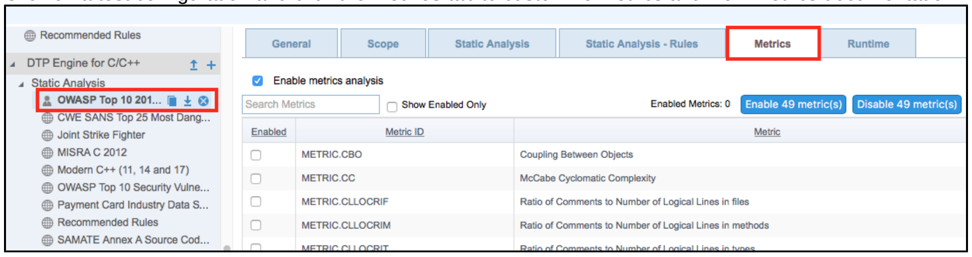
Customizing Metrics Settings

You can configure metrics using **Metrics** tab in the DTP test configuration interface.

1. In DTP, choose **Test Configurations** from the gear icon menu.



2. Click on a test configuration and click the **Metrics** tab to customize metrics and view metrics documentation.



Enable metrics analysis ✓

Show Enabled Only

Enabled Metrics: 0

Enabled	Metric ID	Metric
☐	METRIC.CBO	Coupling Between Objects
☐	METRIC.CC	McCabe Cyclomatic Complexity
☐	METRIC.CLLOCRIF	Ratio of Comments to Number of Logical Lines in files
☐	METRIC.CLLOCRIM	Ratio of Comments to Number of Logical Lines in methods
☐	METRIC.CLLOCRIT	Ratio of Comments to Number of Logical Lines in types

See the DTP manual for details.

Setting Metrics Thresholds

You can set upper and lower boundaries so that a static analysis violation is reported if a metric is calculated outside the specified value range. For example, if you want to restrict the number of logical lines in a project, you could configure the Metrics test configuration so that a violation is reported if the Number of Logical Lines metric exceeds the limit.

The Metrics test configuration shipped with C++test includes default threshold values. There are some rules, such as Number of Files (METRIC.NOF), for which thresholds cannot be set.
Metric thresholds can be set using the following methods:

- By using the test configuration interface in DTP.
- By manually editing the test configuration file:
  1. Go to **Parasoft> Test Configurations> Builtin**, right-click a selected test configuration and choose **Export...** to save the configuration as a . properties file.
  2. Open the exported .properties file in an editor and set the `[METRIC.ID].ThresholdEnabled` property to true.
  3. Configure the lower and upper boundaries in the `[METRIC.ID].Threshold property` according to the following format:

  ```
  [METRIC.ID].Threshold=l [lower boundary value] g [upper boundary value]
  ```

  4. Save the file and go to **Parasoft> Test Configurations.**

  5. Right-click the **User-defined** folder, choose the **Import...** option, browse for the modified test configuration file and click **Open**.

Built-in Metrics

The following metrics are available:

- Coupling Between Objects [METRIC.CBO]
- McCabe Cyclomatic Complexity [METRIC.CC]
- Ratio of Comments to Number of Logical Lines in files [METRIC.CLLOCRIF]
- Ratio of Comments to Number of Logical Lines in methods [METRIC.CLLOCRIM]
- Ratio of Comments to Number of Logical Lines in types [METRIC.CLLOCRIT]
- Nested If Statements [METRIC.DIF]
- Essential Cyclomatic Complexity [METRIC.ECC]
- Fan Out [METRIC.FO]
- Halstead Difficulty [METRIC.HDIFM]
- Halstead Effort [METRIC.HEFM]
- Halstead Intelligent Content [METRIC.HICM]
- Halstead Program Length [METRIC.HLENM]
- Halstead Program Level [METRIC.HLEVM]
- Halstead Number of Bugs [METRIC.HNOBM]
- Halstead Time to Program [METRIC.HTTPM]
- Halstead Program Vocabulary [METRIC.HVOCM]
- Halstead Program Volume [METRIC.HVOLM]
- Inheritance Depth of Class [METRIC.IDOC]
- Lack of Cohesion [METRIC.LCOM]
- Modified Cyclomatic Complexity [METRIC.MCC]
- Maintainability Index [METRIC.MI]
- Nested Blocks Depth [METRIC.NBD]
- Number of Blank Lines in Files [METRIC.NOBLIF]
- Number of Blank Lines in Methods [METRIC.NOBLIM]
- Number of Blank Lines in Types [METRIC.NOBLIT]
- Number of Classes [METRIC.NOC]
- Number of Comment Lines in Files [METRIC.NOCLIF]
- Number of Comment Lines in Methods [METRIC.NOCLIM]
- Number of Comment Lines in Types [METRIC.NOCLIT]
- Number of Files [METRIC.NOF]
- Number of Logical Lines in Files [METRIC.NOLLOCIF]
- Number of Logical Lines in Methods [METRIC.NOLLOCIM]
- Number of Logical Lines in Types [METRIC.NOLLOCIT]
- Number of Methods in Types [METRIC.NOMIT]
- Number of Parameters of Methods [METRIC.NOPAR]
- Number of Physical Lines in Files [METRIC.NOPLIF]
- Number of Physical Lines in Methods [METRIC.NOPLIM]
- Number of Physical Lines in Types [METRIC.NOPLIT]
- Number of Private Members of Types [METRIC.NOPRIVMIT]
- Number of Protected Members of Types [METRIC.NOPROTMIT]
- Number of Public Members of Types [METRIC.NOPUBMIT]
- Number of Returns in Methods [METRIC.NORET]
- Number of Source Lines in Files [METRIC.NOSLIF]
- Number of Source Lines in Methods [METRIC.NOSLIM]
- Number of Source Lines in Types [METRIC.NOSLIT]
- Number of Types [METRIC.NOT]
- Response for Class [METRIC.RFC]
- Strict Cyclomatic Complexity [METRIC.SCC]
- Weighted Methods of Class [METRIC.WMC]