Configuring Static Analysis Settings

By default, DTP considers the violations in different branches as different violations. If you run static analysis on multiple branches using the same run configuration, the same violations will be reported across builds as new violations. As a result, widgets that show change, such as the Violations Trend and Violations - Changed widgets, will present inaccurate information.

You can change the static analysis settings by setting the following JVM argument in JAVA_MEM in the variables file (no file extension). This file is located in the <DTP_HOME>/bin directory:

```
-Ddtp.widgets.compareViolationsAcrossBranches=true
```

When this mode is turned on, DTP will consider violations in different branches as the same violations.

Example Scenario

The following scenario describes the default static analysis settings:

1. Static analysis runs on the master branch (build A)
2. For the following day's build (build B), a release branch is created from the master branch and analyzed with static analysis.
3. On both builds, a static analysis violation is found on Foo.java. The only difference between violation in build A and build B is the branch of the source code.
    - Build A on master branch: Violation 1 on Foo.java
    - Build B on release branch: Violation 1 on Foo.java
4. When comparing build B against build A, the Violations Trend and Violations Changed widgets will show violation 1 in build A as fixed and violation 1 in build B as New.

Changing the static analysis settings as described above will enable the Violations Trend and Violations Changed widgets to report violation 1 in build B as an existing violation when comparing build B against build A.