Modified Coverage 2.0

This DTP Workflow artifact reports the coverage percentage for the lines of code that have been changed since a baseline build and graphically indicates the coverage achieved on each new or modified line of code. It analyzes the coverage data associated with file modifications, determines the lines of code in those files that were changed, then reports which of those new/modified lines of code could be covered, but have not yet been covered.
This helps reduce the testing scope associated with code changes (such as implementing a feature) so testers can focus on the uncovered code and increase their testing efficiency. It also helps teams maintain an audit trail of the risk associated with untested changes.

In this chapter:

Requirements

- Parasoft Extension Designer 5.3.2 or higher
- Parasoft DTP 5.3.2 or higher (see Server Settings)
- The following must be configured in DTP:
  - At least one filter must be configured to receive Run Configurations from coverage analysis runs.
  - Coverage data must be reported to the filter.
  - Access to the source code. You can integrate DTP with your source control system or configure the DTP Engines to publish sources to DTP as part of their analysis (default). For details, see Configuring Source Code Views.
  - A baseline coverage "image" (a.k.a. the baseline build) should be configured so that comparisons can be made against the baseline. See the following section for details.

Definitions

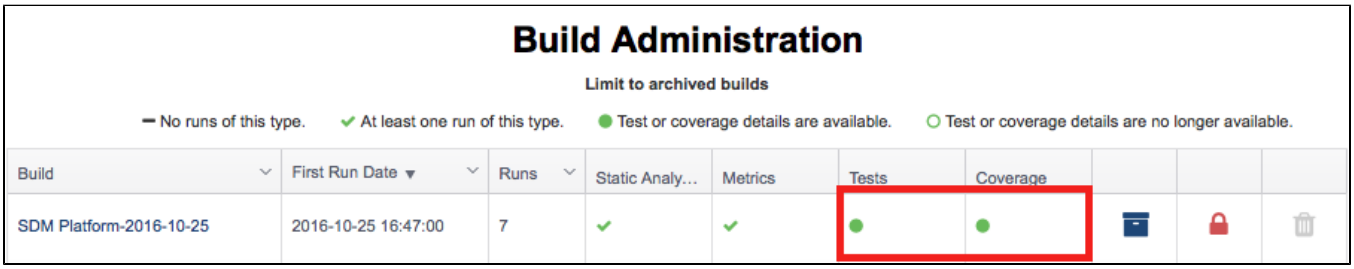
DTP Project	The root level at which the analysis tools report data. Comes preconfigured with a "filter" that gathers all the Run Configurations that are reported to the project.
Filter	A DTP configuration that organizes how the data from different analysis runs is displayed in dashboard widgets and reports.
Run Configuration	A run configuration represents a series of runs (test executions and/or analyses). Runs tagged with the same analysis tool (DTP Engine), DTP project, test configuration, and session tag are grouped into the same run. Run configurations can narrow the data shown for a given filter.
Build ID	Analysis run metadata that is used to aggregate data across multiple analysis runs into a single "build."

Locking Builds

Because DTP continually receives large amounts data, it routinely cleans test data from its database. Generally, the unit test and coverage information associated with a build will be automatically deleted after two more builds containing unit test and coverage information are reported to the same DTP project.

You can lock builds that contain your baseline data to prevent it from being removed. We recommend keeping the baseline build ID locked while using the build for Modified Coverage and/or for DTP audit reports. See Locking and Archiving Builds for details.

You should make sure to lock baseline builds that contain coverage and test data.



Caching the Data

The workflow includes a caching mechanism to speed up multiple requests for the same data. When data is requested, the slice first determines if the data is already computed and cached. If the cache exists, the data is returned directly and the lengthy computation is skipped. The cache is cleared and recomputed on the fly, however, if no data is cached, if the cached data is associated with a different build combination, or if additional analysis data has been reported to the build combination. There is one cache per filter and combination of baseline and target build.

Clearing the Cache

Because the slice does not automatically remove cached data, the cache can grow as more filters are introduced. To help you clear out old cache data, the slice provides a way to delete all cached calculations from the PIE database. This flow cleans all cached calculations. The cache also clears at 00:00 every day. You can configure the auto cache clearing setting by editing the Clean Cache inject node.

Widget Configuration

When the Untested Changes slice is deployed to DTP, you can add the Modified Coverage – Percent widget from the Process Intelligence category to your dashboard (see Adding Widgets).

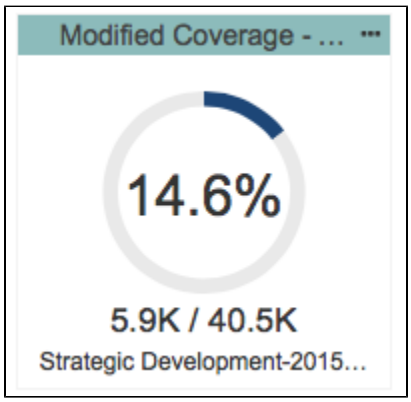
The following settings are available:

Title	Click in the title field to rename the widget (optional).
Filter	Choose Dashboard Settings to use the dashboard filter or choose a filter from the drop-down menu.
Period	Choose Dashboard Settings to use the dashboard period or choose a time range from the drop-down menu.
Baseline Build	Choose a build for comparison. By default, the previous build is selected.
Target Build	Choose the build you want to compare the baseline build to. By default, the latest build is selected.

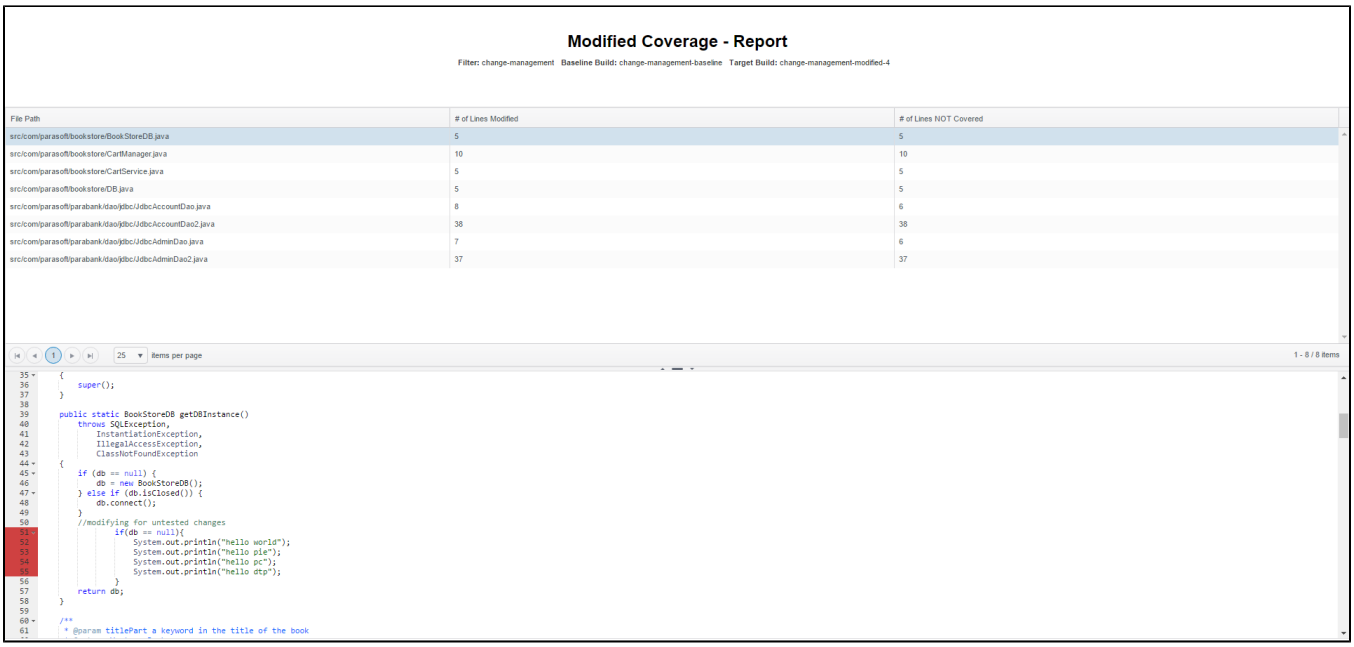
Check Build Administration to Get the Correct Build

By default, Baseline Build is set to Previous Build and Target Build is set to Latest Build. The slice will automatically select the two most recent builds, but these builds may not contain test and coverage details. You should check the Build Administration page in DTP and use an appropriate baseline and target build when configuring the widget as described in the Requirements section. Also see Build Administration.

The widget shows the percentage of modified lines of code that have been covered with tests from the baseline build to the target build.



Click on the widget to open a detailed drill-down report stat shows the coverage for each file:



Red and green are used to highlight testable code that was changed between the baseline build and the target build. Green indicates that at least one test case covered the code. Red indicates that no test cases covered the code.

Reports published to DTP must have consistent information

Instead of presenting code coverage data, the drill-down report may show N/A if one or more of the reports for the builds lacks proper resource information. Verify that the source-related settings for code analysis tool sending reports to DTP have been properly configured. See the tool's documentation for additional information.