Integrating with Polarion ALM

In this section:

Introduction

Polarion ALM is a popular browser-based platform for managing requirements. Parasoft DTP integrates with Polarion ALM, providing the following functionality:

- Ability to manually create defects and issues in Polarion ALM from the Violations Explorer view
- Ability to manually create defects and issues in Polarion ALM from the Test Explorer view.
- Ability to send, view, and update Parasoft test results in Polarion work items (see Sending Test Data to Polarion).
- Traceability from Polarion requirements to tests, static analysis results, and code reviews (see Viewing the Traceability Report).

Requirements

The following requirements are only applicable if you are going to send data to Polarion.

- Tests executed by the following Parasoft tools are supported:
  - C/C++test Professional, dotTEST, or Jtest 10.4.3 +
  - Selenic 2020.1 +
  - SOAtest 9.10.8 +
- You should have already created Polarion work items with one or more of the following types:
  - system requirement
  - software requirement
  - requirement
  - user story

Configuration

The configuration is performed by the Parasoft administrator and only needs to be set up once. Developers, testers, and other DTP end users should review the Usage section for instructions on how to use Parasoft with Polarion ALM.

Connecting DTP to Polarion ALM Server

1. Choose **Report Center Settings** from the settings (gear icon) drop-down menu.
2. Choose **External Application** and choose Polarion from the Application Type drop-down menu.
3. Enable the **Enabled** option.
4. Enter a name for your instance of Polarion in the Name field. The name is required but does not affect the connection settings or render in any other interfaces.
5. Enter the Polarion server in the Application URL field. The URL should include the protocol, host, and port number. Do not include paths or parameters.
6. The Display URL field is rendered in DTP interfaces when links to your Polarion system are created. This URL should include additional paths that may be necessary to access Polarion in a browser
7. Enter login credentials in the Username and Password/API Tokens fields. The login must have sufficient privileges to create issues in the Polarion projects specified in the Project Associations section.
8. Click **Test Connection** to verify your settings and click **Save**.

Associating Parasoft Projects with Polarion Projects

Creating links between Parasoft with Polarion at the project level enables defects created in the Violations or Test Explorer view to be created in the correct project in Polarion.

1. Click **Create Project Association** and choose a project from the DTP Project drop-down menu in the overlay.
2. Enter the name of a Polarion project in the External Project field and click **Create** to save the association.

Click the trash icon to remove a project association. Deleting the project association does not remove links to defects in Polarion from the explorer view. If a new association is created, existing links between violations and Polarion issues will be reactivated.

You can associate multiple projects in DTP with a project in Polarion, but you cannot associate the same DTP project with more than one Polarion project.

Enabling the Requirements Traceability Report

You can configure DTP to generate widgets and reports that help you demonstrate traceability between the requirements stored in Polarion and the test, static analysis, and build review data sent to DTP from Parasoft tools (C/C++test, dotTEST, Jtest, SOAtest).

Your source code files must be associated with work items in Polarion. See Associating Requirements with Files for instructions.

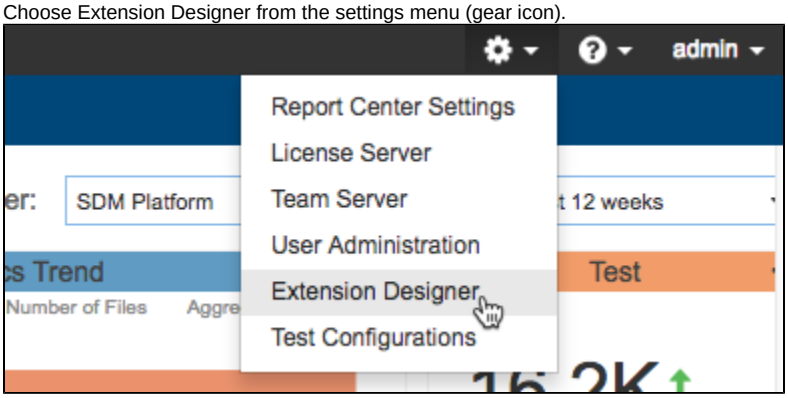
DTP interfaces that display and track traceability are enabled by deploying the External Application Traceability Report artifact shipped with the Traceability Pack. The Traceability Pack also includes the Sending Test Data to External Application flow, which automates part of the requirements traceability workflow. Refer to the Traceability Pack documentation for additional information about the pack.

Use DTP Extension Designer to deploy the External Application Traceability Report and the Sending Test Data to External Application flow to your environment. Verify that DTP is connected to Polarion as described in the Connecting DTP to Polarion ALM Server section before deploying the artifact.

Installing the Traceability Pack

The first step is to install the Traceability Pack. The artifact is a collection of configuration files and assets that enable traceability.

1. Choose Extension Designer from the settings menu (gear icon).

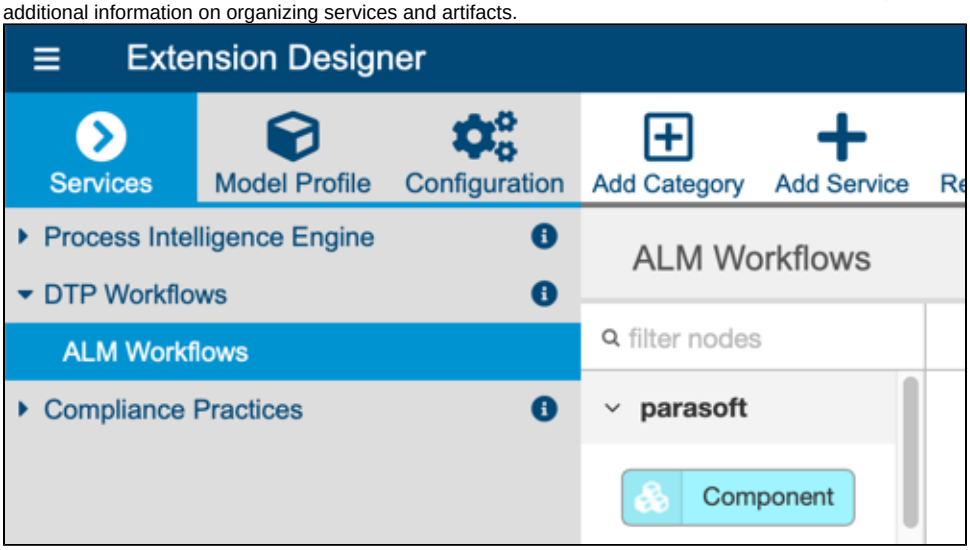


2. Click the **Configuration** tab to open Artifact Manager.
3. Click **Upload Artifact** and browse for the traceability-pack-<version>.zip archive (also see Downloading and Installing Artifacts).
4. Click **Install** and a collection of assets and configuration files for enabling traceability will be installed.

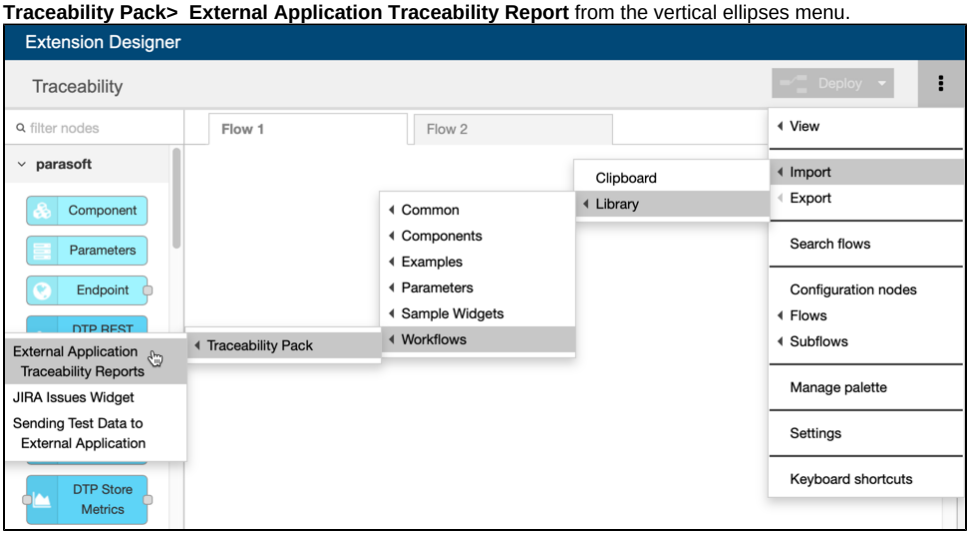
Deploying the External Application Traceability Report

Deploy the External Application Traceability Report after installing the Traceability Pack.

1. Open Extension Designer and click on the **Services** tab.
2. Choose an existing service to deploy the artifact or create a new service in the DTP Workflows category. Refer to Working with Services for additional information on organizing services and artifacts.



3. If you are adding the artifact to an existing service, add a new Flow tab (see Working with Flows) and choose **Import> Library> Workflows> Traceability Pack> External Application Traceability Report** from the vertical ellipses menu.

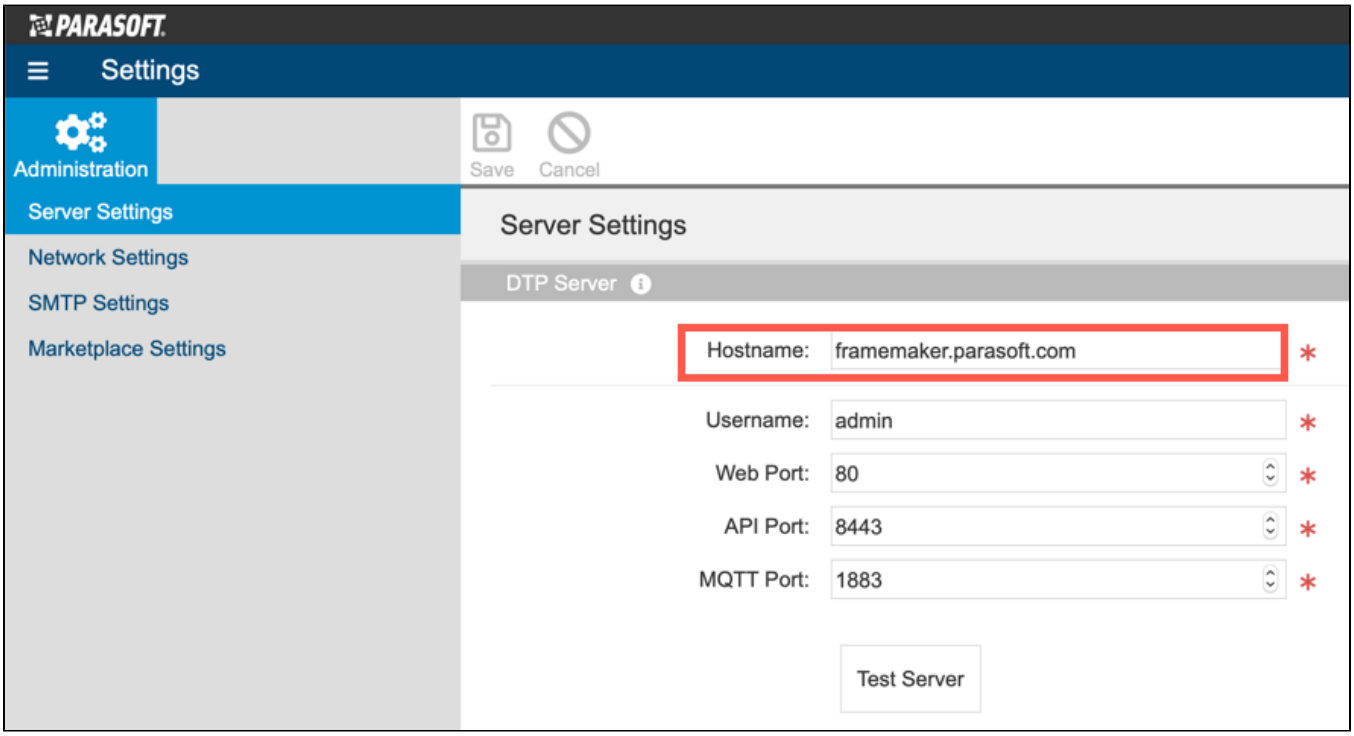


4. Click inside the Flow tab to drop the nodes into the service and click **Deploy**.

Deploying the External Application Traceability Report adds new widgets to Report Center, as well as a drill-down report. See Viewing the Traceability Report for instructions on adding the widgets and viewing the report.

Deploying the Sending Test Data to External Application Flow

This artifact sends test data to Polarion when DTP Data Collector retrieves test results from a Parasoft tool. This artifact ships with the Traceability Pack, which must be installed as described in Installing the Traceability Pack before deploying the flow. You should also verify that the DTP Enterprise Pack connection is to DTP is configured with the host name of the server running DTP.



By default, Enterprise Pack points to `localhost`. See Server Settings for additional information.

1. Open Extension Designer and click on the **Services** tab.
2. Choose an existing service to deploy the artifact or create a new service in the DTP Workflows category. Refer to Working with Services for additional information on organizing services and artifacts.
3. If you are adding the artifact to an existing service, add a new Flow tab (see Working with Flows) and choose **Import> Library> Workflows> Traceability Pack> Sending Test Data to External Application** from the vertical ellipses menu.
4. Click inside the Flow tab to drop the nodes into the service and click **Deploy**.

Usage

After configuring the integration with Polarion, developers, testers, and other users can leverage the functionality enabled by the integration.

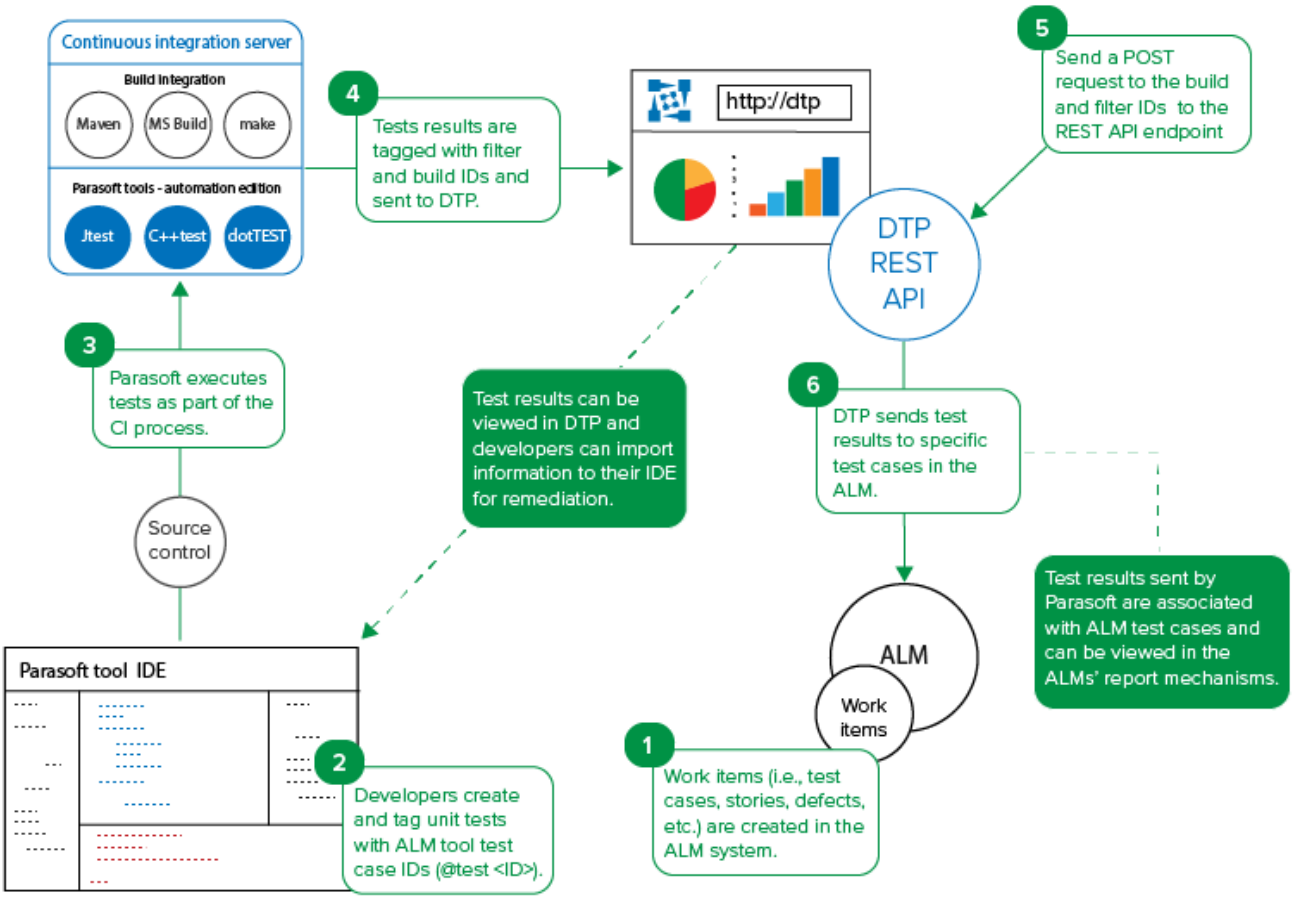
Manually Creating Defects and Issues in Polarion ALM

The Test Explorer and Violations Explorer views enable you to create issues and defects for any test and violation, respectively, regardless of status. Refer to the following sections for details on creating Polarion assets in explorer views:

- See Creating an Issue in a Third-party System for instructions on how to manually create defects and issues in Polarion ALM from the Violations Explorer view.
- See Creating an Issue in a Third-party System for instructions on how to manually create defects and issues in Polarion ALM from the Test Explorer view.

Sending Test Data to Polarion

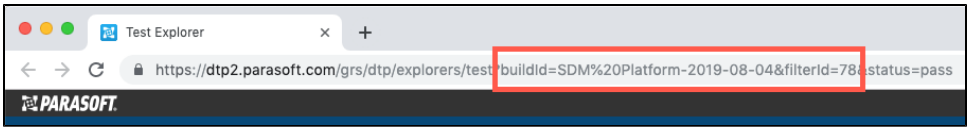
The following diagram shows how you could implement an automated infrastructure for integrating Parasoft DTP and Parasoft test execution tools into your Polarion ALM environment:



1. Create work items in Polarion ALM that will be associated with tests executed by Parasoft C/C++test, dotTEST, or Jtest.
2. In your test file, add the Polarion test case or requirement IDs using the `@test` or `@req` annotation. See the C/C++test, dotTEST, or Jtest documentation for details on adding annotations.
    - Use the `@test <Polarion unit test case ID>` annotation in your tests to associate them with Polarion unit test cases.
    - Use the `@req <Polarion software/system requirement ID>` annotation in your tests to associate them with Polarion software or system requirements.
3. Execute your tests as part of the CI process. You can also manually execute the tests from the C/C++test desktop.
4. As part of the test execution, C/C++test will tag the results with the filter and build IDs and send the data to DTP. You can verify the results in DTP by adding Test Widgets to your DTP dashboard and setting the filter and build ID. C/C++test developers can download the test execution data from DTP into their IDEs so that they can address any failed tests.
5. If you deployed the Sending Test Data to External Application flow (see Deploying the Sending Test Data to External Application Flow), then unit and functional testing results will automatically be sent to Polarion when Data Collector receives the data from the Parasoft tool.
   You can also manually send a POST request to the DTP REST API endpoint to send results from the DTP database to Polarion. Pass the DTP filter and build IDs as URL parameters in the API call:

```
curl -X POST -u <username>:<password> "http://<host>:<port>/grs/api/v1.7/linkedApps/configurations/1
/syncTestCases?filterId=<filterID>&buildId=<buildID>"
```

The filter and build IDs are available in the Test Explorer URL:



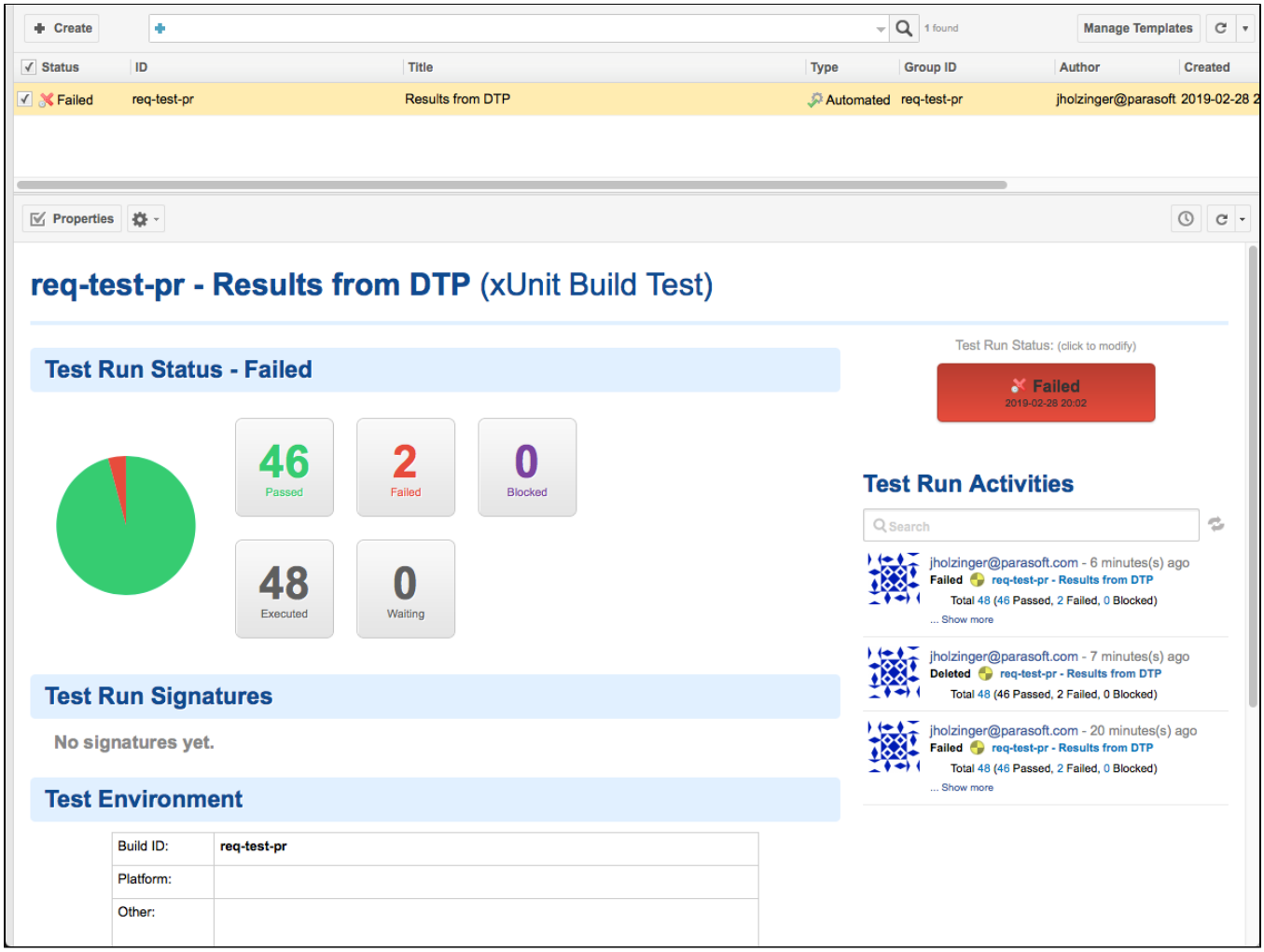
6. DTP will locate the test results that match the `filterId` and `buildId` parameters and send the data to the Polarion unit test cases or requirements. You should expect the following response:
    - When DTP locates results with an `@test <ID>`, it will search for unit test cases with a matching ID in Polarion and update the item. No action will be taken if the unit test case IDs do not exist in Polarion.
    - When DTP locates results with an `@req <ID>`, it will search for requirements with a matching ID in Polarion and update associated children unit test cases. If no unit test cases exist for the requirement IDs, unit test cases will be created. Unit test cases will also be created if the requirement IDs are not found.
    - An external-app-sync.log file will also be written to the the <DTP_INSTALL>/logs directory. This log file contains progress information about sending test results from DTP to Polarion.

After DTP processes the report and sends results to Polarion, you should expect a response similar to the following:

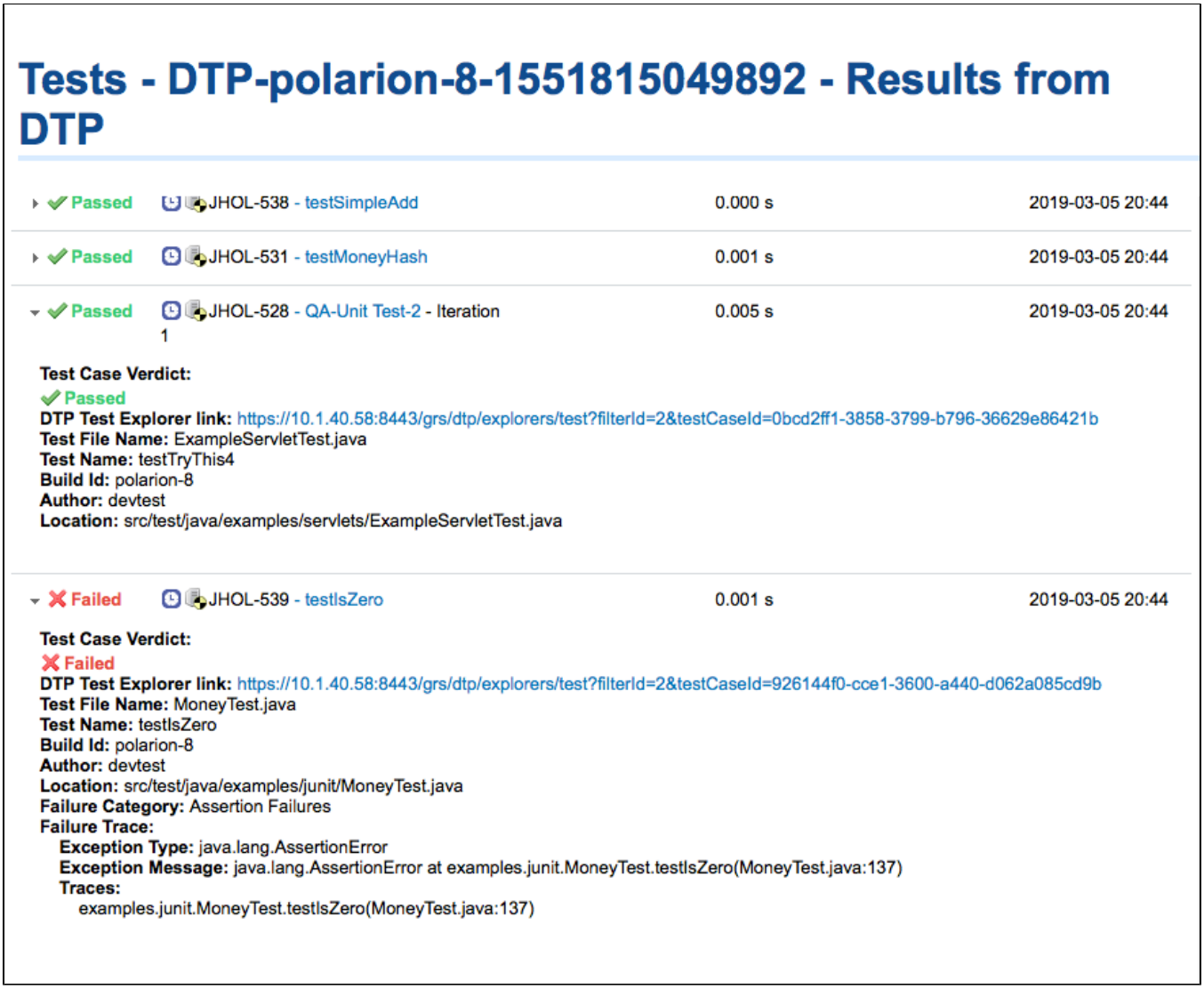
```
{
  "createdTestSession": "DTPP-521",
    "created": [
        "DTPP-519, testName = testBagSumAdd"
    ],
    "updated": [
        "DTPP-519, testName = testBagSumAdd",
        "DTPP-518, testName = testBagSimpleAdd"
    ],
    "ignored": [
        "MAGD-567, testName = testBagNegate",
        "QAP-512, testName = testTryThis3",
        "QAP-512, testName = testTryThis4",
        "MAGD-567, testName = testBagMultiply"
    ]
}
```

Viewing Results in Polarion

After successfully sending the test data to Polarion, you will be able to view results in Polarion.



You can drill down into Polarion reports to view additional details about the tests, including authorship, location, and execution time. Refer to the Polarion documentation for details about understanding reports in Polarion.



Tests - DTP-polarion-8-1551815049892 - Results from DTP

Passed	JHOL-538 - testSimpleAdd	0.000 s	2019-03-05 20:44
Passed	JHOL-531 - testMoneyHash	0.001 s	2019-03-05 20:44
Passed	JHOL-528 - QA-Unit Test-2 - Iteration 1	0.005 s	2019-03-05 20:44

Test Case Verdict:
Passed
DTP Test Explorer link: https://10.1.40.58:8443/grs/dtp/explorers/test?filterId=2&testCaseId=0bcd2ff1-3858-3799-b796-36629e86421b
Test File Name: ExampleServletTest.java
Test Name: testTryThis4
Build Id: polarion-8
Author: devtest
Location: src/test/java/examples/servlets/ExampleServletTest.java

Failed	JHOL-539 - testIsZero	0.001 s	2019-03-05 20:44

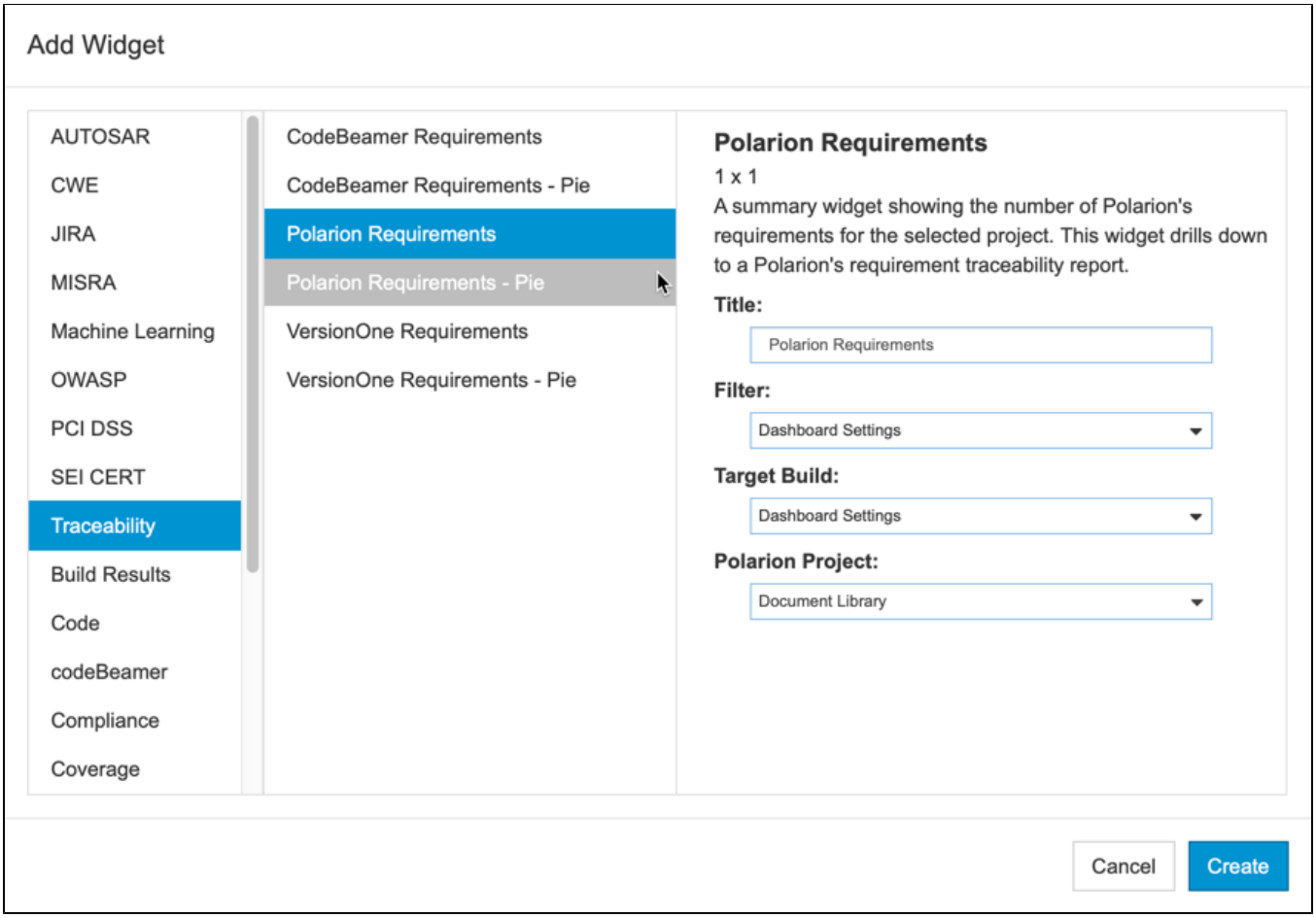
Test Case Verdict:
Failed
DTP Test Explorer link: https://10.1.40.58:8443/grs/dtp/explorers/test?filterId=2&testCaseId=926144f0-cce1-3600-a440-d062a085cd9b
Test File Name: MoneyTest.java
Test Name: testIsZero
Build Id: polarion-8
Author: devtest
Location: src/test/java/examples/junit/MoneyTest.java
Failure Category: Assertion Failures
Failure Trace:
Exception Type: java.lang.AssertionError
Exception Message: java.lang.AssertionError at examples.junit.MoneyTest.testIsZero(MoneyTest.java:137)
Traces:
examples.junit.MoneyTest.testIsZero(MoneyTest.java:137)

Viewing the Traceability Report

If the External Application Traceability Report has been deployed to your system (see Enabling the Requirements Traceability Report), you can add widgets to your dashboard to monitor traceability from requirements to tests, static analysis, code reviews for your project. The widgets also drill down to a report that includes additional details.

Adding and Configuring the Widgets

The widgets will appear in a separate Traceability category when adding widgets to your DTP dashboard. See Adding Widgets for general instructions on adding widgets.

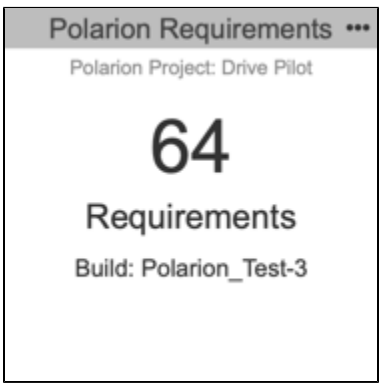


You can configure the following settings:

Title	You can enter a new title to replace the default title that appears on the dashboard.
Filter	Choose Dashboard Settings to use the dashboard filter or choose a filter from the drop-down menu. See Creating and Managing Filters for additional information about filters.
Target Build	This should be set to the build ID you executed the tests and code analysis under. You can use the build specified in the dashboard settings, the latest build, or a specific build from the drop-down menu. Also see Configuring Dashboard Settings.
Type	*Pie widget only*. Choose either a Tests, Violations, or Reviews from the drop-down menu to show a pie chart detailing the status by type. Add instances of the widget configured to each type for a complete overview in your dashboard.
Project	Choose a Polarion project from the drop-down menu.

Requirements Widget

This widget shows the number of requirements from the specified Polarion project.



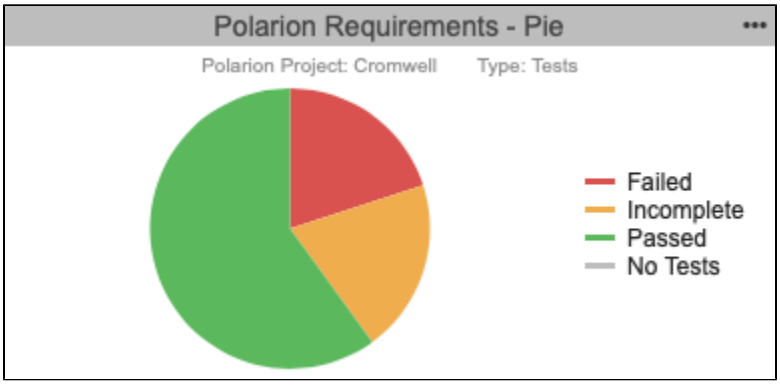
Click on the widget to open the Requirement Traceability report.

Pie Widget

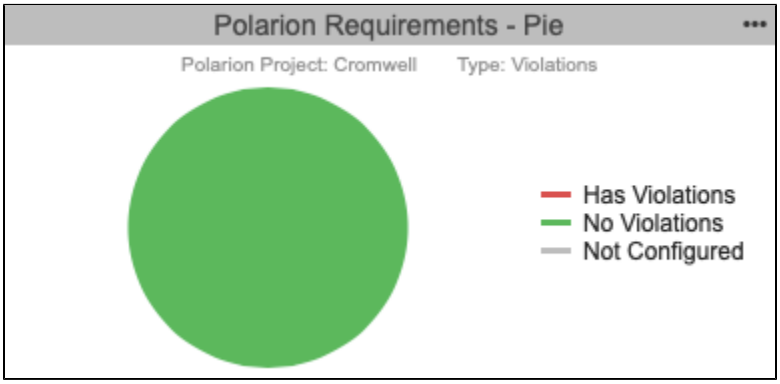
Unit testing, functional testing, static analysis, and peer reviews are common activities for verifying that requirements have been properly and thoroughly implemented. This widget shows the overall status of the project requirements in the context of those software quality activities. You can add a widget for each type of quality activity (tests, static analysis violations, reviews) to monitor the progress of requirements implementation for the project.

Mouse over a section of the chart to view details about quality activity type status. Click on the widget to open the Requirement Traceability report filtered by the selected type.

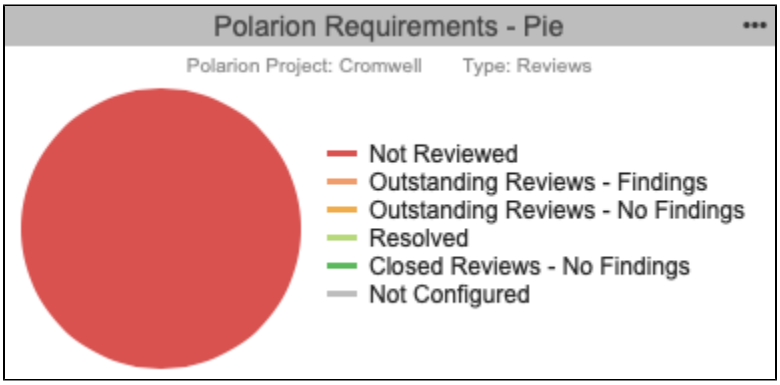
Requirements Implementation Status by Tests



Requirements Implementation Status by Violations

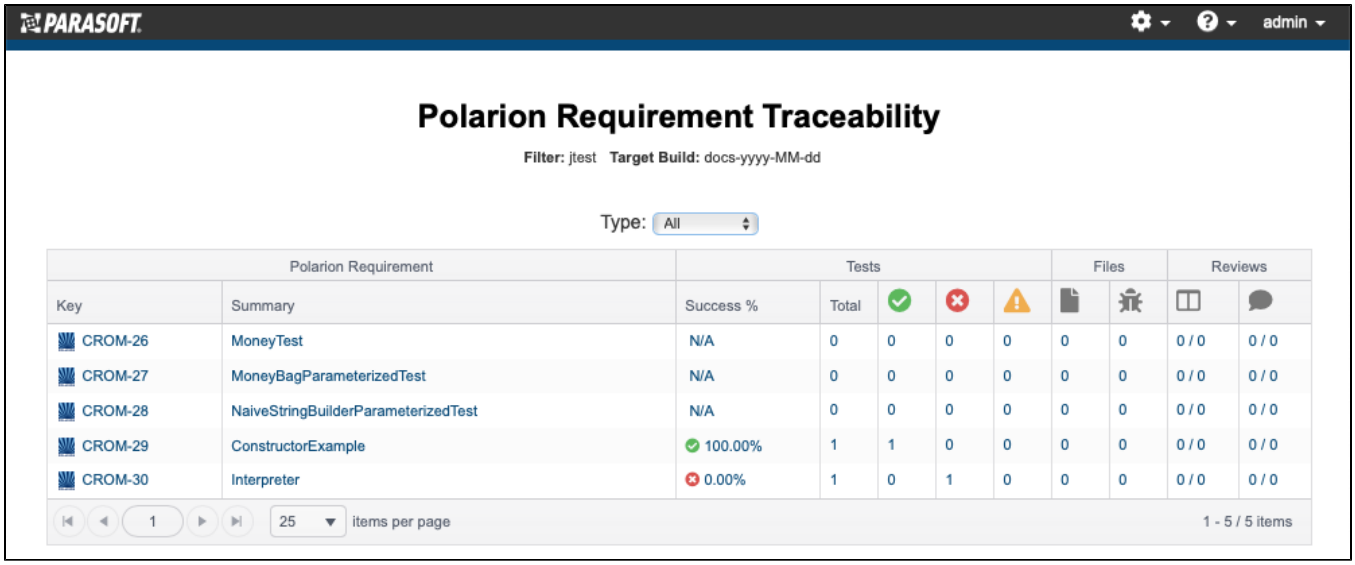


Requirements Implementation by Reviews



Understanding the Requirement Traceability Report

The report lists the Polarion project requirements and data associated with them.

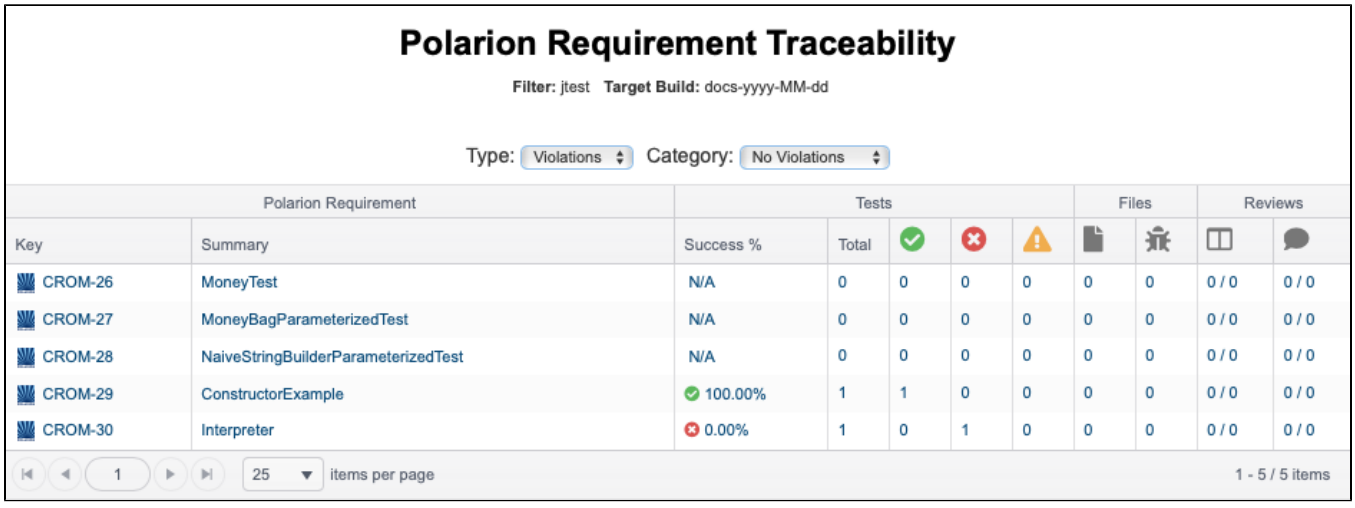


You can perform the following actions:

- Click on a link in the Key column to view the requirement in Polarion.
- Click on a link in the Summary column or one of the Test columns to view the test-related information associated with the requirement in the Requirement Details Report.
- Click on a link in one of the Files columns to view the static analysis-related information associated with the requirement in the Requirement Details Report.
- Click on a link in one of the Reviews columns to view the change review-related information associated with the requirement in the Requirement Details Report.

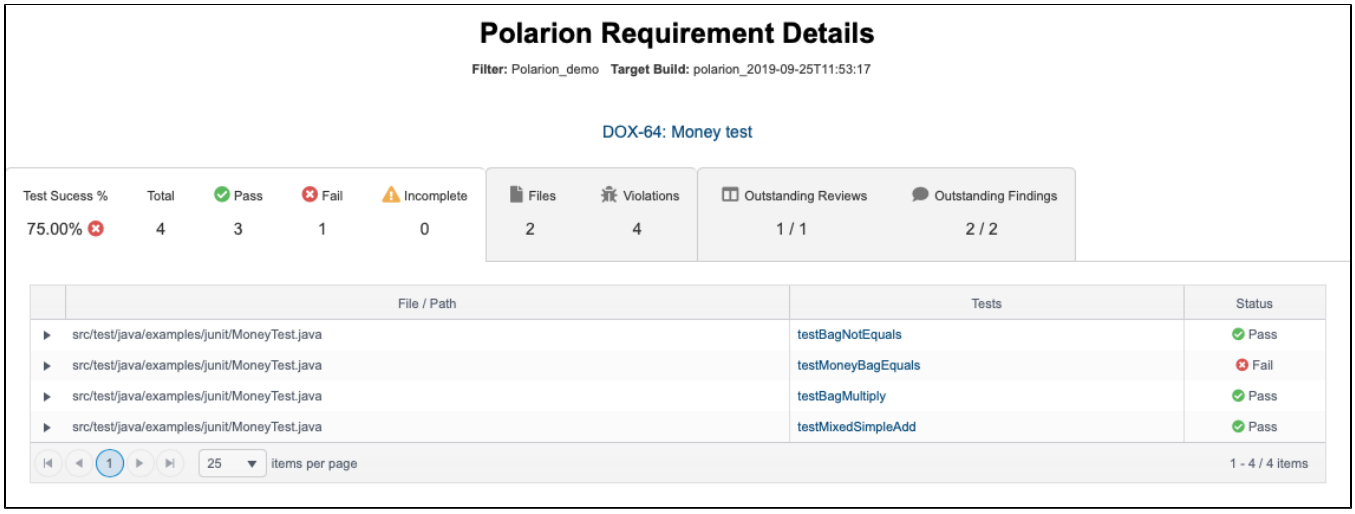
Requirement Traceability Report by Type

Clicking on a section of the Requirements - Pie widget opens a version of the report that includes only the quality activity type selected in the widget. You can use the drop-down menus to switch type and status.

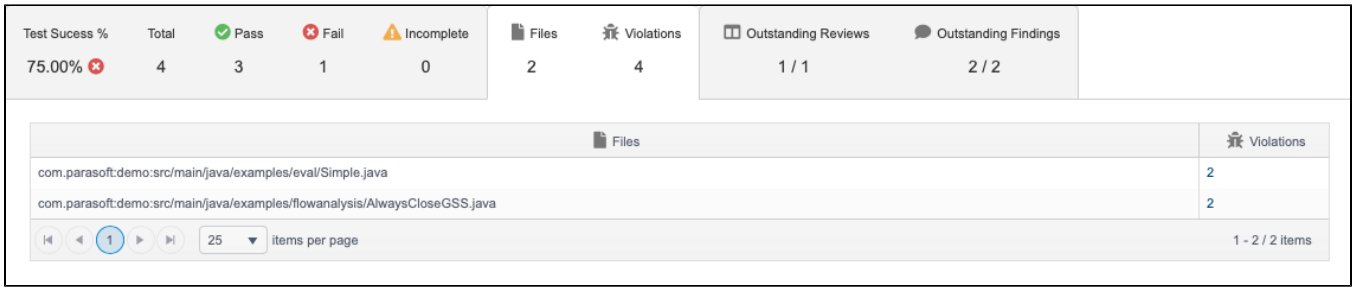


Understanding the Requirement Details Report

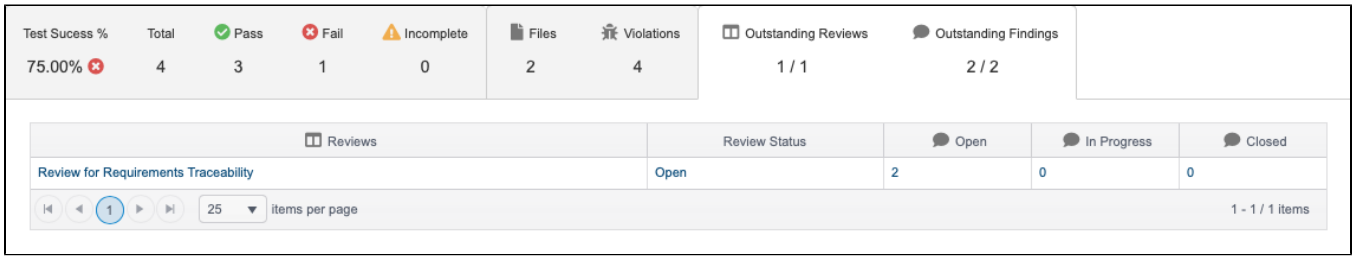
The Requirement Details report provides additional information about the files, static analysis findings, and tests associated with a specific Polarion requirement. You can open this report by clicking on a requirement in the main Requirement Traceability report.



The first tab shows the results of the tests that were executed to verify the specific requirement. Click on a test name to view the test in the Test Explorer.



The second tab shows the files associated with the specific requirement, as well as the static analysis violations detected in the files. You can click the link the Violations column to view the violations in the Violations Explorer, which provides additional details about the violations.



If the files include any change reviews or review findings, they will be shown in the third tab with links to view them in the Change Explorer.