

Integrating with Polarion ALM

In this section:

- [Introduction](#)
- [Requirements](#)
- [Configuration](#)
- [Usage](#)

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 System** and choose Polarion from the System 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

Associating a Parasoft project with a Polarion project enables you to create defects from the Violations or Test Explorer views that are linked to the correct project in Polarion. The association is also important when using the [Sending Test Data to External System flow](#). 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.

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 reconfigure an existing association between DTP and Polarion projects:

1. Click the pencil icon and choose either a different DTP project from the drop-down menu or specify the name of a different Polarion project in the External Project field.

2. Click **Save**.

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).

If you want the Traceability Report to include code review and static analysis information, you must associate your source code files with work items in Polarion. See [Associating Requirements with Files](#) for instructions on enabling this optional feature.

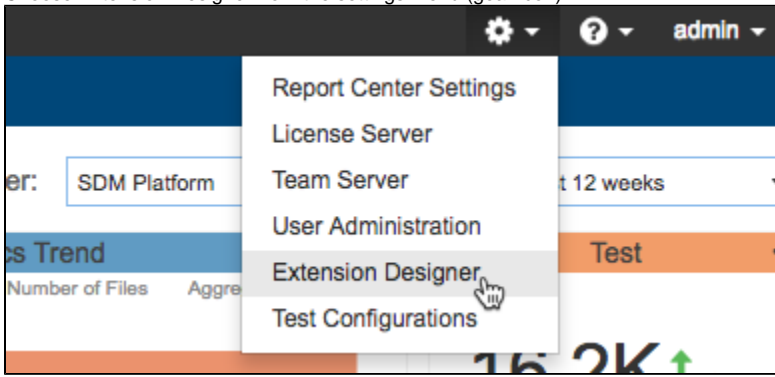
DTP interfaces that display and track traceability are enabled by deploying the External System Traceability Report artifact shipped with the Traceability Pack. The Traceability Pack also includes the Sending Test Data to External System flow, which automates part of the requirements traceability workflow.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

Use DTP Extension Designer to deploy the External System Traceability Report and the Sending Test Data to External System 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).

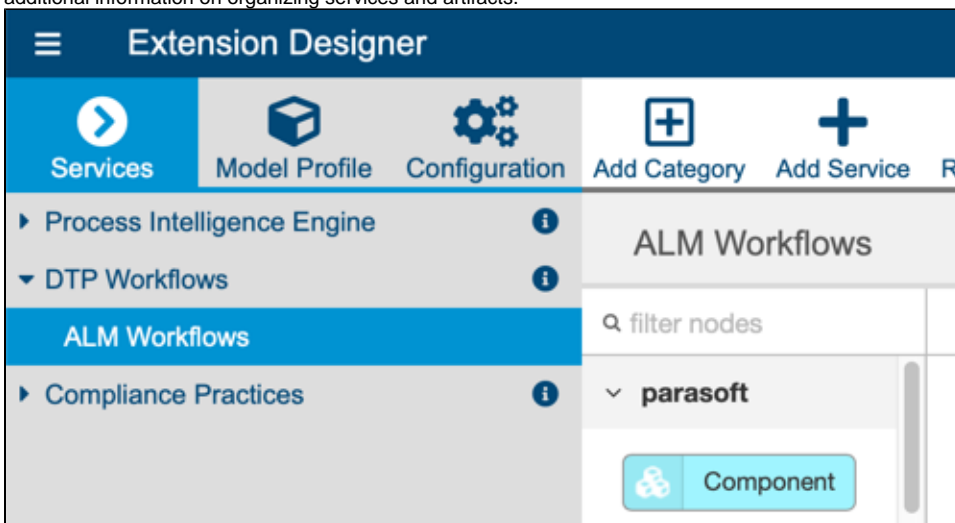


2. Click the **Configuration** tab to open Artifact Manager.
3. Click **Upload Artifact** and browse for the traceability-pack-<version>.zip archive (also see [Downloading and Installing Artifacts](#)).
4. Click **Install** and a collection of assets and configuration files for enabling traceability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

Deploying the External System Traceability Report

Deploy the External System 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** from the ellipses menu.
4. Choose **Library> Workflows> Traceability Pack> External System Traceability Report** and click **Import**.
5. Click inside the Flow tab to drop the nodes into the service and click **Deploy**.

Deploying the External System 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 System 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.

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 from the ellipses menu**.
4. Choose **Library > Workflows > Traceability Pack > Sending Test Data to External System Application** and click **Import**.
5. Click inside the Flow tab to drop the nodes into the service and click **Deploy**.

Advanced Configuration

You can modify the *ExternalSystemSettings.properties* configuration file located in the <DTP_DATA_DIR>/conf directory to change the default behavior of the integration. DTP's out-of-the-box integration with Polarion is configured to use default or commonly-used fields and work item types. If you customized your Polarion system, however, then you can configure the following settings to align data in DTP with your custom configuration.

polarion.workItemType.requirementIds	<p>Specifies a comma-separated list of Polarion work item types that should take the role of requirements in Parasoft. The work items are also used in the Traceability Report.</p> <p>Choose Polarion Administration > Work Items > Types for the IDs in the Polarion UI to access the work item IDs in your Polarion system.</p> <p>Default:</p> <pre>polarion.workItemType.requirementIds=softwarerequirement,systemrequirement,requirement,userstory</pre>
polarion.workItemType.testId	<p>Specifies the Polarion work item type that should take the role of tests in Parasoft.</p> <p>Choose Polarion Administration > Work Items > Types for the IDs in the Polarion UI to access the work item IDs in your Polarion system.</p> <p>Default:</p> <pre>polarion.workItemType.testId=unittestcase</pre>
polarion.workItemType.issue	<p>Specifies the type of work item to create in Polarion when creating new issues from the DTP Violation Explorer and Test Explorer. This enables you to associate custom issue trackers you may have configured in Polarion with work items created from DTP.</p> <p>By default, the property is not set. As a result, issue work items created in DTP are associated with issue work items in Polarion.</p>
polarion.workItemType.task	<p>Specifies the type of work item to create in Polarion when creating new tasks from the DTP Violation Explorer and Test Explorer. This enables you to associate custom task trackers you may have configured in Polarion with work items created from DTP.</p> <p>By default, the property is not set. As a result, task work items created in DTP are associated with task work items in Polarion.</p>
polarionIssueUrl	<p>Specifies the URL template for linking work items created in the DTP Violation Explorer and Test Explorer to work items in Polarion.</p> <p>Default:</p> <pre>polarionIssueUrl=<POLARION_URL>/polarion/redirect/project/<PROJECT>/workitem?id=<ID></pre>
polarion.hyperlinkRole	<p>Specifies the role of hyperlinks as configured in your Polarion system. The hyperlink role is used when creating work items from the DTP Violations Explorer and Test Explorer.</p> <p>Choose Polarion Administration > Enumerations > hyperlink-role-enum.xml in Polarion to access the hyperlink role.</p> <p>Default:</p> <pre>polarion.hyperlinkRole=ref_ext</pre>
polarion.requirementStatus.verified	<p>Specifies the relationship between the tests sent from DTP and the requirements in Polarion. Choose Polarion Administration > Enumerations > workitem-link-role-enum.xml in Polarion to view the current relationship configurations.</p> <p>Default:</p> <pre>polarion.requirementStatus.verified=verifies</pre>

Usage

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

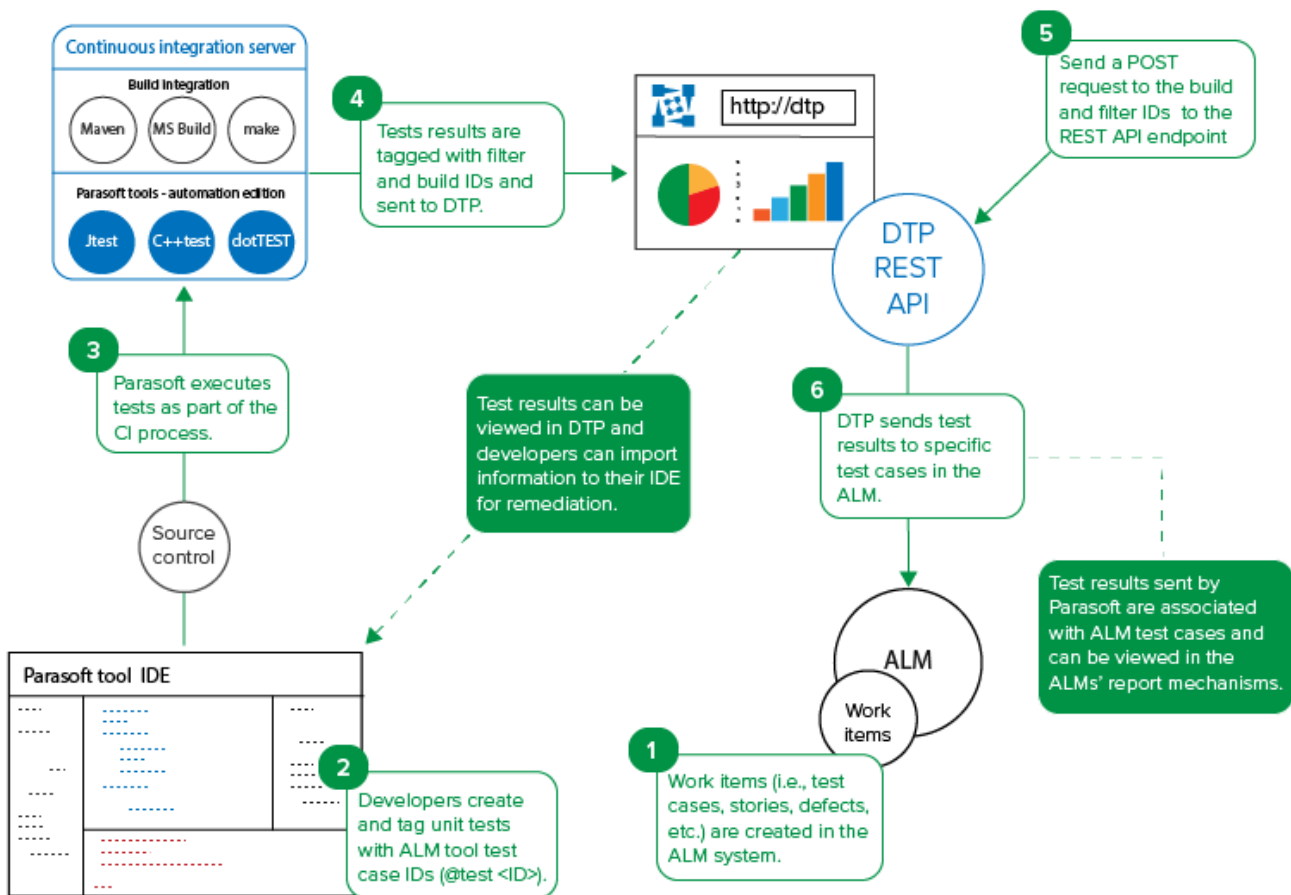
Manually Creating Defects and Issues in Polarion ALM

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.

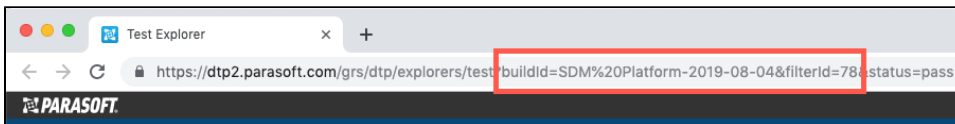
- The Polarion unit test case or requirement ID is available in several Polarion interfaces, such as the URL:



- Execute your tests as part of the CI process. You can also manually execute the tests from the C/C++test desktop.
- 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.
- If you deployed the Sending Test Data to External System flow (see [Deploying the Sending Test Data to External System Flow](#)), then unit and functional testing results will automatically be sent to Polarion when Data Collector receives the data from the Parasoft tool. By default, the flow forwards unit and functional test results that were received by Data Collector for any project, but you can configure the flow to only send data for a specific project (see [Sending Results from a Specific DTP Project](#)). 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:



- 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 `<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:

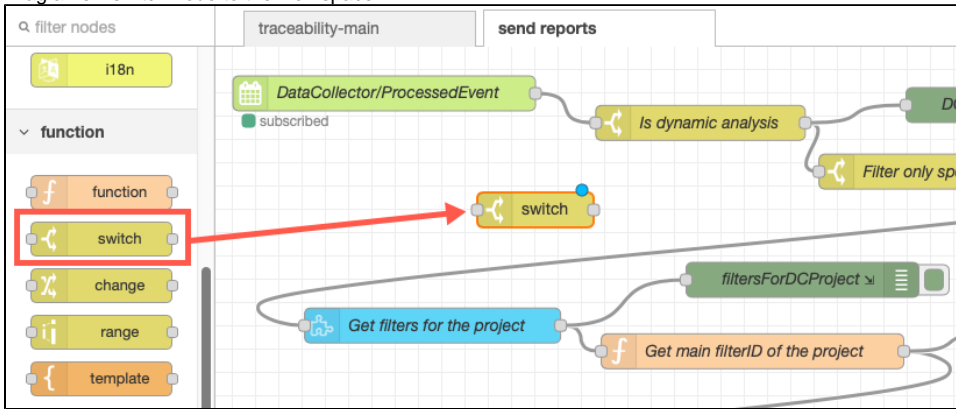
```
{
  "createdTestSession": "DTPP-521",
  "created": [
    "DTPP-519, testName = testBagSumAdd"
  ],
  "updated": [
    "DTPP-519, testName = testBagSumAdd",
    "DTPP-518, testName = testBagSimpleAdd"
  ],
  "ignored": [
    "MAGD-567, testName = testBagNegate",
    "QAP-512, testName = testTryThis3",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

Sending Results from a Specific DTP Project

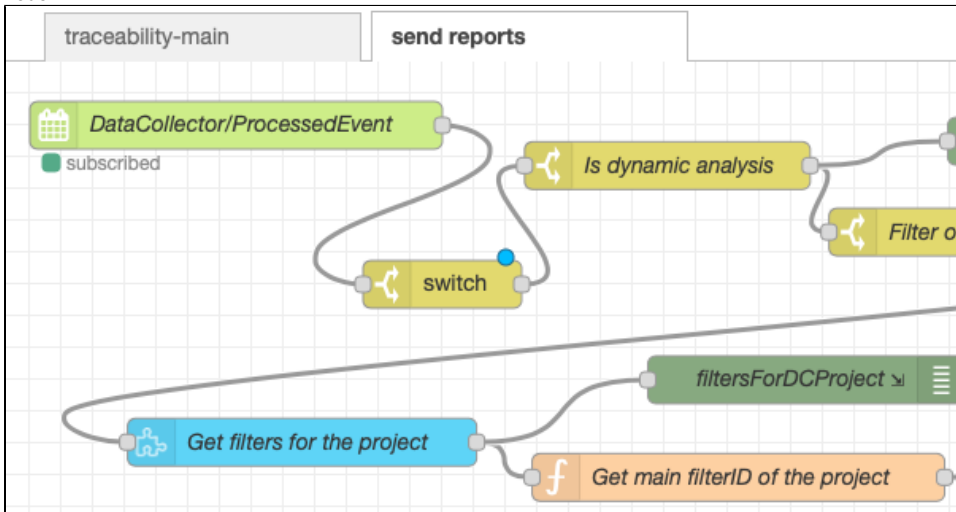
If you are using the Sending Test Data to External System flow to forward unit and functional test results, data will be sent to Polarion for all DTP projects by default. As a result, work items will be updated to include the tests collected for any DTP project that contain annotations matching Polarion IDs. You can configure the flow, however, to only send data for a specific project.

1. Open Extension Designer and open the service where the Sending Test Data is deployed.

2. Drag a new switch node to the workspace.



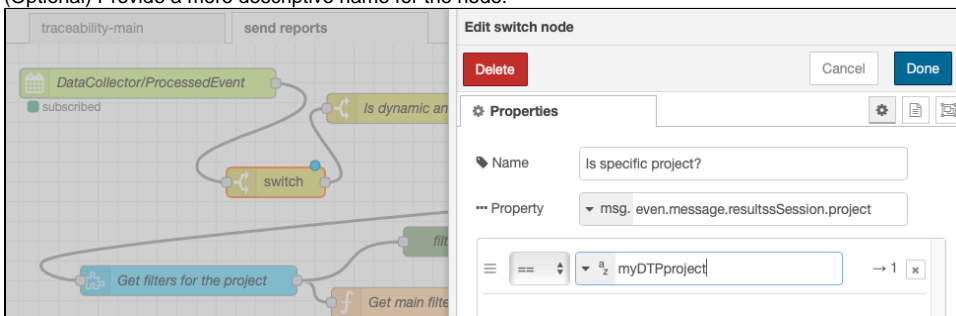
3. Select and delete the connector line between the "DataCollectorProcessedEvent" node and the "Is dynamic analysis" node.
4. Drag a new connector from the "DataCollectorProcessedEvent" node to the switch node and from the switch node to the "Is dynamic analysis" node.



5. Double-click the node and specify the following string in the Property field:

```
event.message.resultsSession.project
```

6. Specify the name of the DTP project in the string field.
7. (Optional) Provide a more descriptive name for the node.



8. Click **Done** to finish configuring the node and click **Deploy**.

When the flow executes, only test results for the specified DTP project will be sent to Polarion.

Viewing Results in Polarion

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

[+ Create](#)
1 found
Manage Templates

Status	ID	Title	Type	Group ID	Author	Created
Failed	req-test-pr	Results from DTP	Automated	req-test-pr	jholzinger@parasoft	2019-02-28 20:02

[Properties](#)
⏪ ⏩

req-test-pr - Results from DTP (xUnit Build Test)

Test Run Status - Failed

46 Passed	2 Failed	0 Blocked
48 Executed	0 Waiting	

Test Run Status: (click to modify)

Failed
2019-02-28 20:02

Test Run Activities

- jholzinger@parasoft.com - 6 minutes(s) ago
Failed req-test-pr - Results from DTP
 Total 48 (46 Passed, 2 Failed, 0 Blocked)
[... Show more](#)

- jholzinger@parasoft.com - 7 minutes(s) ago
Deleted req-test-pr - Results from DTP
 Total 48 (46 Passed, 2 Failed, 0 Blocked)

- jholzinger@parasoft.com - 20 minutes(s) ago
Failed req-test-pr - Results from DTP
 Total 48 (46 Passed, 2 Failed, 0 Blocked)
[... Show more](#)

Test Run Signatures

No signatures yet.

Test Environment

Build ID:	req-test-pr
Platform:	
Other:	

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 System 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.

Add Widget

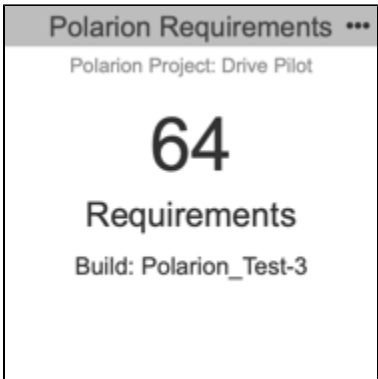
CWE	JIRA Requirements - Pie	<h4>Polarion Requirements</h4> <p>1 x 1</p> <p>A summary widget showing the number of Polarion's requirements for the selected project. This widget drills down to a Polarion's requirement traceability report.</p> <p>Title:</p> <input type="text" value="Polarion Requirements"/> <p>Filter:</p> <input type="text" value="Dashboard Settings"/> <p>Target Build:</p> <input type="text" value="Dashboard Settings"/> <p>Polarion Project:</p> <input type="text"/>
SEI CERT	JIRA Requirements	
Traceability	Jira Test Coverage	
Build Results	Polarion Requirements	
Code	Polarion Requirements - Pie	
Compliance	Polarion Test Coverage	
Coverage	TeamForge Requirements	
Diagnostics	TeamForge Test Coverage	
Metrics	TeamForge Workitems - Pie	
Process Intelligence	VersionOne Test Coverage	
Static Analysis	VersionOne Workitems	
Tests	VersionOne Workitems - Pie	
Custom	codeBeamer Requirements	
	codeBeamer Requirements - Pie	

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	<i>Pie widget only.</i> 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

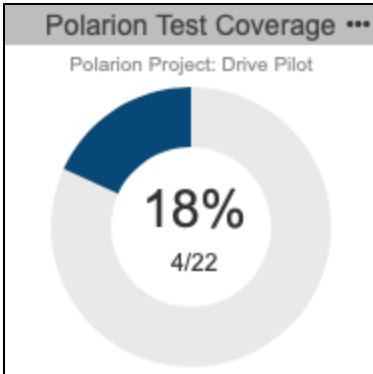
This widget shows the number of requirements from the specified Polarion project.



Click on the widget to open the [Requirement Traceability report](#).

Test Coverage Widget

This widget shows the percentage of requirements covered by tests against all requirements in the project.



Click the center of the widget to open the main [Requirement Traceability report](#).

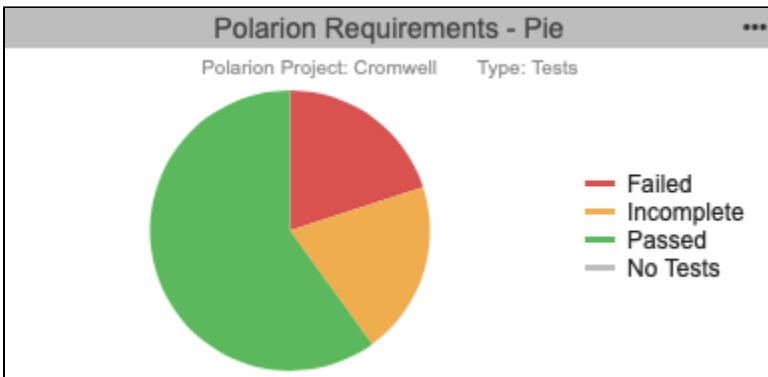
The colored-in segment represents the requirements covered by tests. Click on the segment to open the [Requirement Traceability report](#) filtered to the **With Tests** category.

Pie Widget

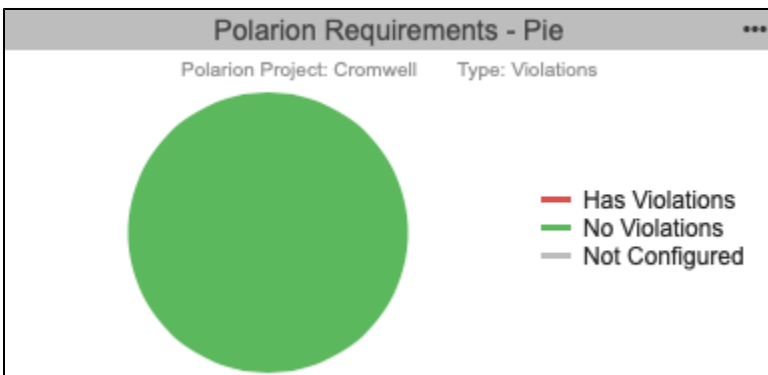
Unit testing, functional testing, static analysis, and peer reviews are common activities for verifying that requirements have been properly and thoroughly implemented. This widget shows the overall status of the project requirements in the context of those software quality activities. You can add a 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.

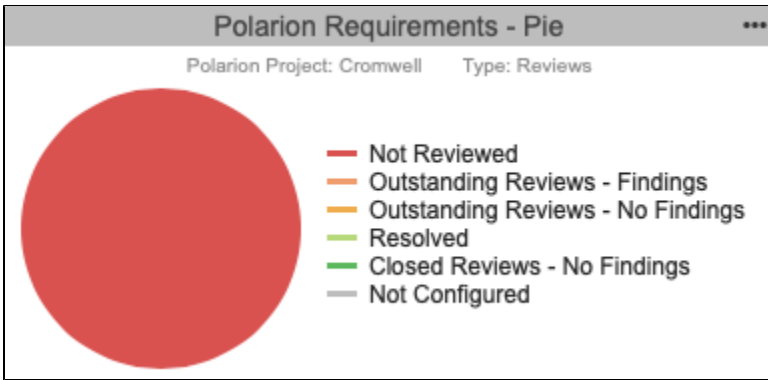
Requirements Implementation Status by Tests



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.

PARASOFT

admin

Polarion Requirement Traceability

Filter: jtest Target Build: docs-yyyy-MM-dd

Type: All Show files/reviews

Polarion Requirement		Tests				Files		Reviews	
Key	Summary	Success %	Total	✓	✗	⚠	📄	👤	🗨
🔗 CROM-26	MoneyTest	N/A	0	0	0	0	0	0	0 / 0
🔗 CROM-27	MoneyBagParameterizedTest	N/A	0	0	0	0	0	0	0 / 0
🔗 CROM-28	NaiveStringBuilderParameterizedTest	N/A	0	0	0	0	0	0	0 / 0
🔗 CROM-29	ConstructorExample	100.00%	1	1	0	0	0	0	0 / 0
🔗 CROM-30	Interpreter	0.00%	1	0	1	0	0	0	0 / 0

25 items per page 1 - 5 / 5 items

You can perform the following actions:

- Disable or enable the **Show files/reviews** option if you want to hide the Files and Reviews columns in the report. The Files and Reviews columns will only contain data if the requirements have been mapped to source files (see [Enabling the Requirements Traceability Report](#)). Disabling the Files and Reviews columns on this screen hides the related tabs in the [Requirement Details report](#).
- 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. You can also disable or enable the **Show files/reviews** option if you want to hide the Files and Reviews columns in the report. The Files and Reviews columns will only contain data if the requirements have been mapped to source files files (see [Enabling the Requirements Traceability Report](#)). Disabling the Files and Reviews columns on this screen hides the related tabs in the [Requirement Details report](#).

Polarion Requirement Traceability

Filter: jtest Target Build: docs-yyyy-MM-dd

Type: Violations Category: No Violations Show files/reviews

Polarion Requirement		Tests				Files		Reviews		
Key	Summary	Success %	Total							
CROM-26	MoneyTest	N/A	0	0	0	0	0	0	0 / 0	0 / 0
CROM-27	MoneyBagParameterizedTest	N/A	0	0	0	0	0	0	0 / 0	0 / 0
CROM-28	NaiveStringBuilderParameterizedTest	N/A	0	0	0	0	0	0	0 / 0	0 / 0
CROM-29	ConstructorExample	100.00%	1	1	0	0	0	0	0 / 0	0 / 0
CROM-30	Interpreter	0.00%	1	0	1	0	0	0	0 / 0	0 / 0

1 - 5 / 5 items

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.

Polarion Requirement Details

Filter: Polarion_demo Target Build: polarion_2019-09-25T11:53:17

DOX-64: Money test

Test Success %	Total	Pass	Fail	Incomplete	Files	Violations	Outstanding Reviews	Outstanding Findings	View results in Test Explorer
75.00%	4	3	1	0	2	4	1 / 1	2 / 2	

File / Path	Tests	Status
▶ src/test/java/examples/junit/MoneyTest.java	testBagNotEquals	Pass
▶ src/test/java/examples/junit/MoneyTest.java	testMoneyBagEquals	Fail
▶ src/test/java/examples/junit/MoneyTest.java	testBagMultiply	Pass
▶ src/test/java/examples/junit/MoneyTest.java	testMixedSimpleAdd	Pass

1 - 4 / 4 items

The first tab shows the results of the tests that were executed to verify the specific work item.

You can click on the **View results in Test Explorer** link to view all of the tests associated with the requirement in the [Test Explorer](#).

You can also click on individual test names in the table to view each test in the Test Explorer.

Test Success %	Total	Pass	Fail	Incomplete	Files	Violations	Outstanding Reviews	Outstanding Findings
75.00%	4	3	1	0	2	4	1 / 1	2 / 2

Files	Violations
com.parasoft:demo:src/main/java/examples/eval/Simple.java	2
com.parasoft:demo:src/main/java/examples/flowanalysis/AlwaysCloseGSS.java	2

1 - 2 / 2 items

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.

This tab will only contain data if the requirements have been mapped to source files files (see [Enabling the Requirements Traceability Report](#)). If you did not map requirements to files, you can hide this tab by disabling the **Show files/reviews** option on the main traceability report page and reloading the details report.

Test Success %	Total	Pass	Fail	Incomplete	Files	Violations	Outstanding Reviews	Outstanding Findings
75.00%	4	3	1	0	2	4	1 / 1	2 / 2

Reviews	Review Status	Open	In Progress	Closed
Review for Requirements Traceability	Open	2	0	0

25 items per page
 1 - 1 / 1 items

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](#).

This tab will only contain data if the requirements have been mapped to source files files (see [Enabling the Requirements Traceability Report](#)). If you did not map requirements to files, you can hide this tab by disabling the **Show files/reviews** option on the main traceability report page and reloading the details report.