Integrating with Jira

In this section:

Introduction

If your organization tracks and manages software development projects in Jira, you can connect DTP to your Jira system in the administration page. Connecting DTP to Jira provides the following functionality:

- Ability to manually create a Jira issue from the Violations Explorer view.
- Ability to manually create a Jira issue from the Test Explorer view.
- If you are using the Xray plug-in for Jira, you can also send, view, and update Parasoft Test results in Jira (see Sending Test Data to Jira/Xray)
- Traceability from Jira requirements to tests, static analysis results, and code reviews (see Viewing the Traceability Report).

Requirements

This feature has been tested with Jira Project Management Software v8.0.2#800010 and Jira Cloud. The feature may not function as expected on other versions of Jira.

The following requirements are only applicable if you are going to send test results to Jira:

- 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 +
- Xray plug-in

Configuration

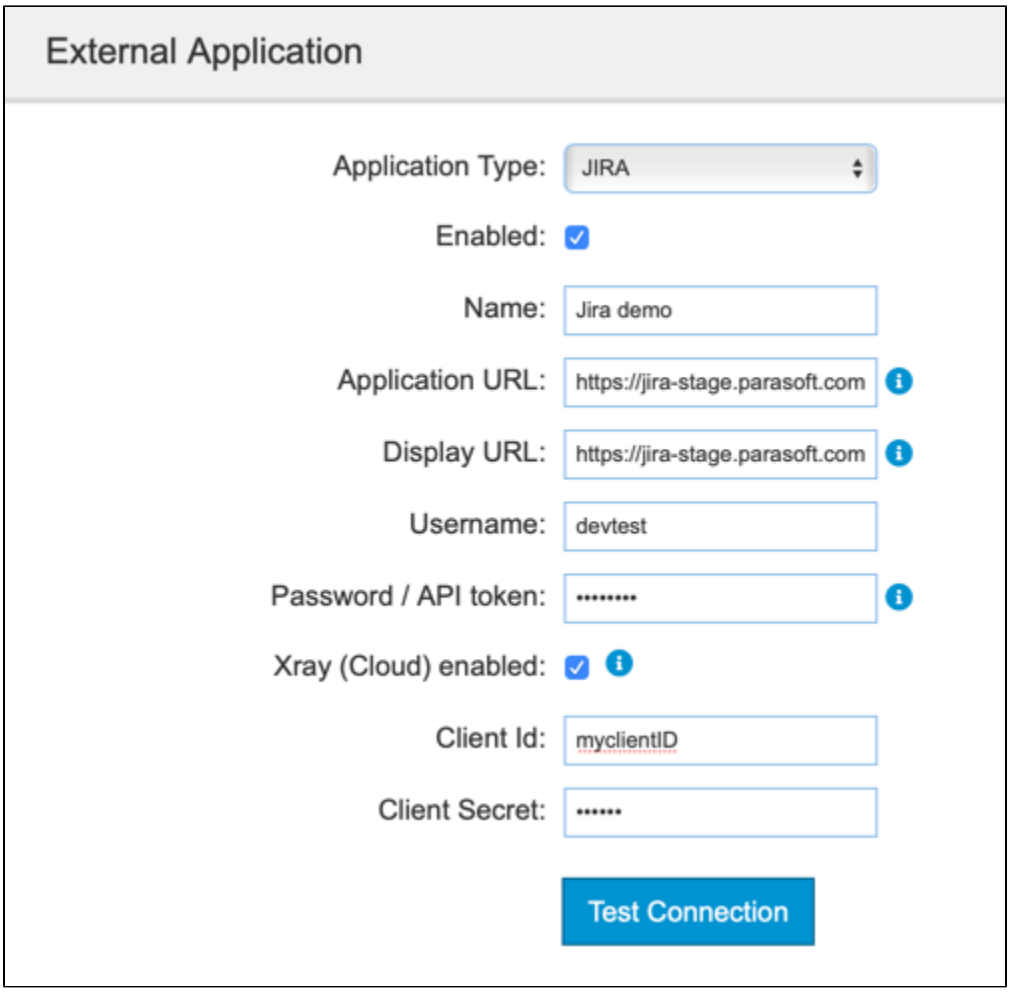
The configuration is performed by the Parasoft administrator and only needs to be set up once. Developers, testers, and other DTP end users should review the Usage section for instructions on how to use Parasoft with Jira.

Connecting DTP to Jira

1. Choose **Report Center Settings** from the settings (gear icon) drop-down menu.
2. Choose **External Application** and choose Jira from the Application Type drop-down menu.
3. Enable the **Enabled** option.
4. Enter a name for the server in the Name field. The name is required but does not affect the connection settings or render in any other interfaces.
5. Enter the URL of your Jira system in the Application URL field. The URL should include the protocol, host, and port number. Do not include paths or parameters, e.g., `http://jira.yourcompany.com:8080`.
6. Enter a URL for rendering links to your Jira system from DTP in the Display URL field. This URL should include additional paths that may be necessary to access Jira in a browser, e.g., `http://jira.yourcompany.com`.
7. Enter your username and either a password (self-managed Jira server) or an API token (Jira Cloud) in the appropriate fields. The login must have sufficient privileges to create Jira issues in the projects specified in the Project Associations section.
8. Click **Test Connection** to verify your settings and click **Save**.

Connecting to Xray Cloud

Xray is an extension for Jira that manages tests. Xray is available as a hosted or cloud-based application. DTP can send data to hosted instances of Xray through the connection as described in Connecting DTP to Jira. For cloud-based instances of Xray, enable the **Xray (Cloud) enabled** option and specify the Xray client ID and secret so that you can leverage the functionality described in the Sending results from DTP to JIRA/Xray section.



Refer to the Xray documentation for information about generating client IDs and secrets: https://confluence.xpand-it.com/display/XRAYCLOUD/Global+Settings%3A+API+Keys

Associating Parasoft Projects with Jira

Create links between Parasoft and Jira projects so that defects created in the Violations or Test Explorer view are created in the correct project in JIra. The association is also important when using the the Sending Test Data to External Application flow.

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 Jira 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 in DTP explorer views to defects in Jira. If an association is deleted and recreated later, existing links between violations and Jira issues will be reactivated.

You can associate multiple projects in DTP with a project in Jira, but you cannot associate the same DTP project with more than one Jira 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 Jira 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 stories in Jira. See Associating Requirements with Files for instructions on enabling this optional feature.

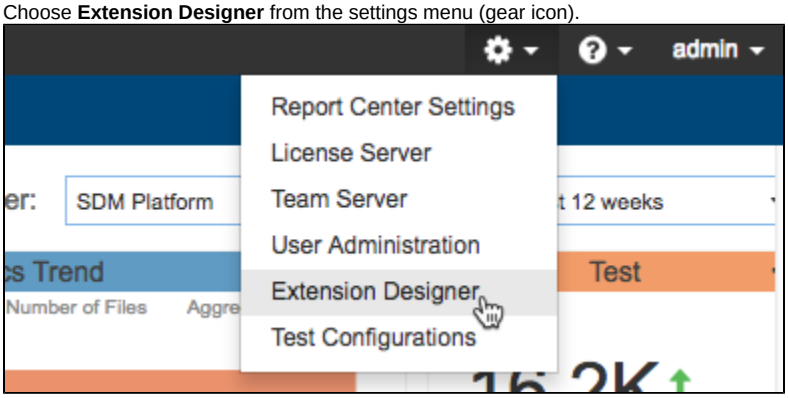
DTP interfaces that display and track traceability are enabled by deploying the External Application Traceability Report artifact shipped with the Traceability Pack. The Traceability Pack also includes the **Sending Test Data to External Application flow**, which automates part of the requirements traceability 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 Jira as described in Connecting DTP to Jira before deploying the artifact.

Installing the Traceability Pack

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

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

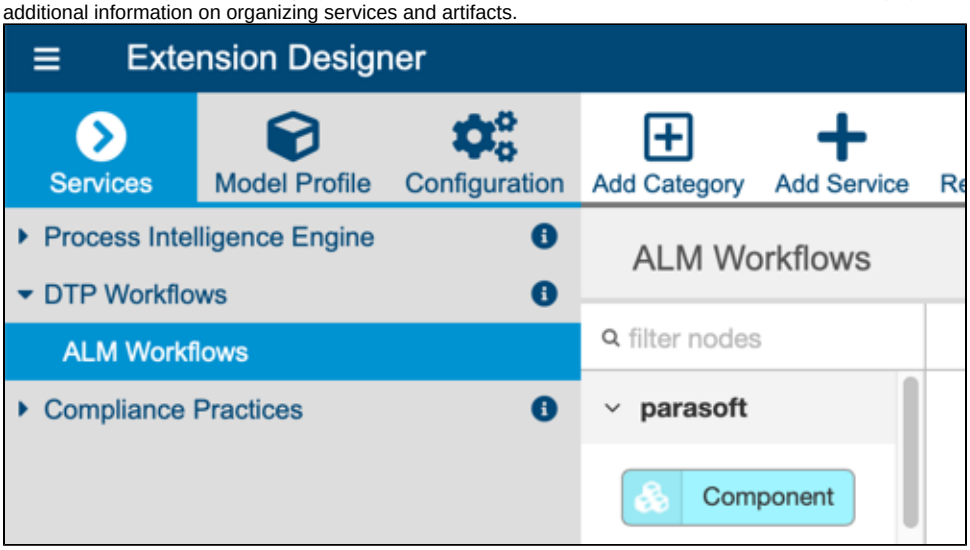


2. Click the **Configuration** tab to open Artifact Manager.
3. Click **Upload Artifact** and browse for the traceability-pack-<version>.zip archive (also see Downloading and Installing Artifacts).
4. Click **Install** and a collection of assets and configuration files for enabling traceability will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

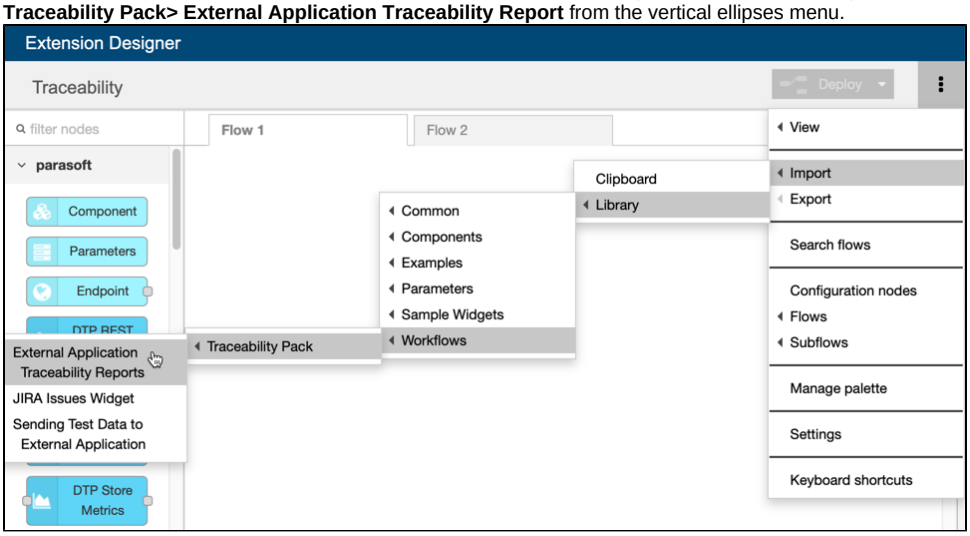
Deploying the Traceability Report

Deploy the report components to your DTP environment after installing the External Application Traceability Report artifact.

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 Jira 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> 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**.

Advanced Configuration

You can modify the ExternalAppsSettings.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 Jira/Xray is configured to use default or commonly-used fields and types. If you customized your Jira /Xray system, however, then you can configure the following settings to align data in DTP with your custom Jira/Xray configuration.

jiraIssueUrl	Specifies the URL template for linking work items created in the DTP Violation Explorer and Test Explorer to work items in Jira. Default: `JiraIssueUrl=<JIRA_URL>/browse/<ID>` The value of the <JIRA_URL> segment of the URL path is specified when connecting DTP to Jira. See Connecting DTP to Jira.
jira.issueType.bug	Specifies the name for the issue type in Jira that takes the role of bugs when creating work items from the DTP Violations Explorer and Test Explorer. By default, the property is not set. As a result, bug work items created in DTP are associated with bug work items in Jira.
jira.issueType.task	Specifies the name for the issue type in Jira that takes the role of tasks when creating work items from the DTP Violations Explorer and Test Explorer. By default, the property is not set. As a result, task work items created in DTP are associated with task work items in Jira.
jira.issueType.test	Specifies the name for the issue type in Jira that takes the role of tests. When test results are sent from DTP to Jira/Xray, tests will be labeled as the issue type specified in this setting. Default: `jira.issueType.test=Test` This setting can be used in on-premise and cloud-based instances of Jira/Xray.
jira.issueType.requirement	Specifies the name for the issue type in Jira that takes the role of requirements. When test results are sent from DTP to Jira/Xray, the requirements associated with the tests will be labeled as the issue type specified in this setting. The issue type is also used to identify requirements in the Traceability Report. Default: `jira.issueType.requirement=Story` This setting can be used in on-premise and cloud-based instances of Jira/Xray.
jira.testDone.transitionId	Specifies a custom status for tests sent from DTP to Jira/Xray. If not specified, the default system status for completed tests is used. See Getting Default Test and Test Execution Statuses for information about getting the default test status names used in your system.
jira.testExecutionDone.transitionId	Specifies a custom status for test executions sent from DTP to Jira/Xray. If not specified, the default system status for test executions is used. See Getting Default Test and Test Execution Statuses for information about getting the test execution status names used in your system.

jira.testType.customfieldId	Specifies the ID of the test type custom field in your Jira/Xray system. The value should match the existing custom field. Tests will be labeled according to the type specified with the jira.testType.customfieldValue setting. Both the `jira.testType.customfieldId` *and* `jira.testType.customfieldValue` settings must be configured to specify a custom value. By default, both `jira.testType.customfieldId` *and* `jira.testType.customfieldValue` settings are not configured. As a result, the default test type value is used. See Getting Values for Custom Test Type Fields for additional details. This setting can be used in on-premise and cloud-based instances of Jira/Xray.
jira.testType.customfieldValue	Specifies which test type is assigned to tests created when results are sent from DTP to Jira/Xray. The value should match the existing custom field in your Jira/Xray system. The custom value is identified by the ID specified in with the `jira.testType.customfieldId` setting. Both the `jira.testType.customfieldId` *and* `jira.testType.customfieldValue` settings must be configured to specify a custom value. If only the `jira.testType.customfieldId` *or* `jira.testType.customfieldValue` settings are configured, will result in an error. By default, both `jira.testType.customfieldId` *and* `jira.testType.customfieldValue` settings are not configured. As a result, the default test type value is used. See Getting Values for Custom Test Type Fields for additional details. This setting can be used in on-premise and cloud-based instances of Jira/Xray.
jira.xrayOnPremises.testCaseFindingStatus.pass	Specifies the test run status name in Jira/Xray to assign to passing test results sent from DTP. This enables you to set custom statuses you may have configured in Jira for test results in DTP. Default: `jira.xrayOnPremises.testCaseFindingStatus.pass=PASS` This setting can be used in on-premise instances of Jira/Xray.
jira.xrayOnPremises.testCaseFindingStatus.fail	Specifies the test run status name in Jira/Xray to assign to failed test results sent from DTP. This enables you to set custom statuses you may have configured in Jira for test results in DTP. Default: `jira.xrayOnPremises.testCaseFindingStatus.fail=FAIL` This setting can be used in on-premise instances of Jira/Xray.
jira.xrayOnPremises.testCaseFindingStatus.incomplete	Specifies the test run status name in Jira/Xray to assign to passing test results sent from DTP. This enables you to set custom statuses you may have configured in Jira for test results in DTP. Default: `jira.xrayOnPremises.testCaseFindingStatus.incomplete=ABORTED` This setting can be used in on-premise instances of Jira/Xray.
jira.xrayOnPremises.testToRequirementRelationName	Specifies the name of the requirement type in Jira for tests sent from DTP. This enables you to associate tests with requirements using a custom name you may have configured in Jira. Default: `jira.xrayOnPremises.testToRequirementRelationName=Tests` This setting can be used in on-premise instances of Jira/Xray.
jira.xrayCloud.testCaseFindingStatus.pass	Specifies the test run status name in Jira/Xray to assign to passing test results sent from DTP. This enables you to set custom statuses you may have configured in Jira for test results in DTP. Default: `jira.xrayCloud.testCaseFindingStatus.pass=PASS` This setting can be used in Cloud instances of Jira/Xray.

jira.xrayCloud.testCaseFindingStatus.fail	Specifies the test run status name in Jira/Xray to assign to passing test results sent from DTP. This enables you to set custom statuses you may have configured in Jira for test results in DTP. Default: `jira.xrayCloud.testCaseFindingStatus.fail=FAIL` This setting can be used in Cloud instances of Jira/Xray.
jira.xrayCloud.testCaseFindingStatus.incomplete	Specifies the test run status name in Jira/Xray to assign to passing test results sent from DTP. This enables you to set custom statuses you may have configured in Jira for test results in DTP. Default: `jira.xrayCloud.testCaseFindingStatus.incomplete=ABORTED` This setting can be used in Cloud instances of Jira/Xray.
jira.xrayCloud.testToRequirementRelationName	Specifies the name of the requirement type in Jira for tests sent from DTP. This enables you to associate tests with requirements using a custom name you may have configured in Jira. Default: `jira.xrayCloud.testToRequirementRelationName=Test` This setting can be used in Cloud instances of Jira/Xray.

Getting Values for Custom Test Type Fields

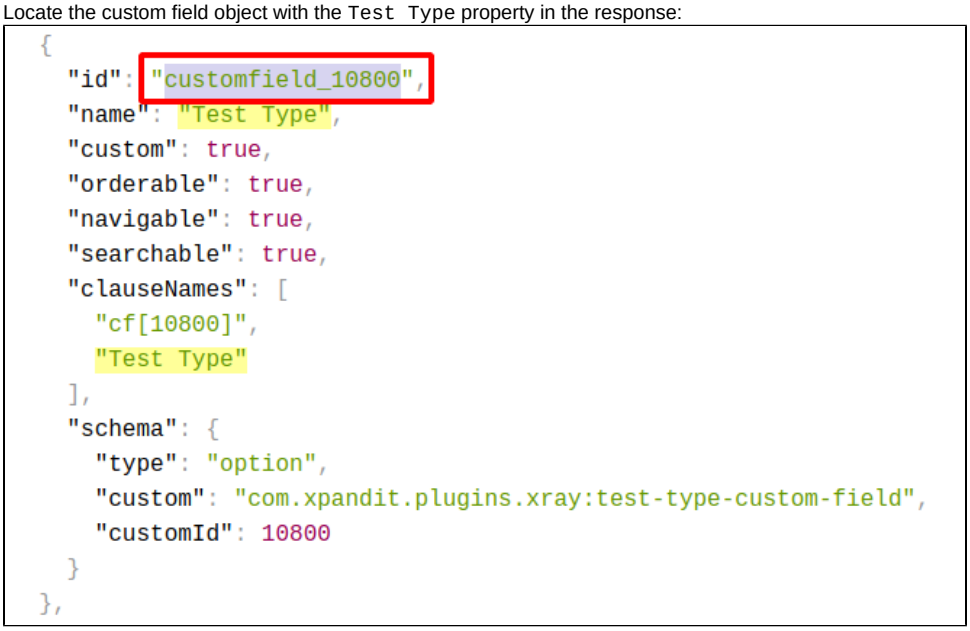
The `jira.testType.customfieldId` and `jira.testType.customfieldValue` settings are used to specify a custom type for tests sent to Jira /Xray from DTP. The `jira.testType.customfieldId` setting identities the name of the field for the custom type, and the `jira.testType.customfieldValue` setting identifies the specific value. Both values must exist in your Jira/Xray system.

To get the value for the `jira.testType.customfieldId` setting:

1. Send a GET request to the following endpoint to get the value of the custom field:

```
curl -X GET "http(s)://<jira-host>/rest/api/2/field"
```

2. Locate the custom field object with the `Test Type` property in the response:



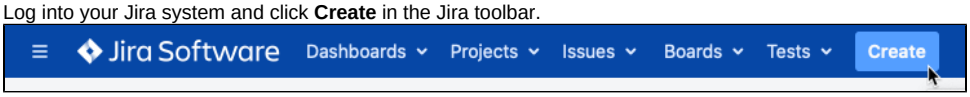
```
{
  "id": "customfield_10800",
  "name": "Test Type",
  "custom": true,
  "orderable": true,
  "navigable": true,
  "searchable": true,
  "clauseNames": [
    "cf[10800]",
    "Test Type"
  ],
  "schema": {
    "type": "option",
    "custom": "com.xpandit.plugins.xray:test-type-custom-field",
    "customId": 10800
  }
},
```

3. Set the `jira.testType.customfieldId` setting to the `id` property:

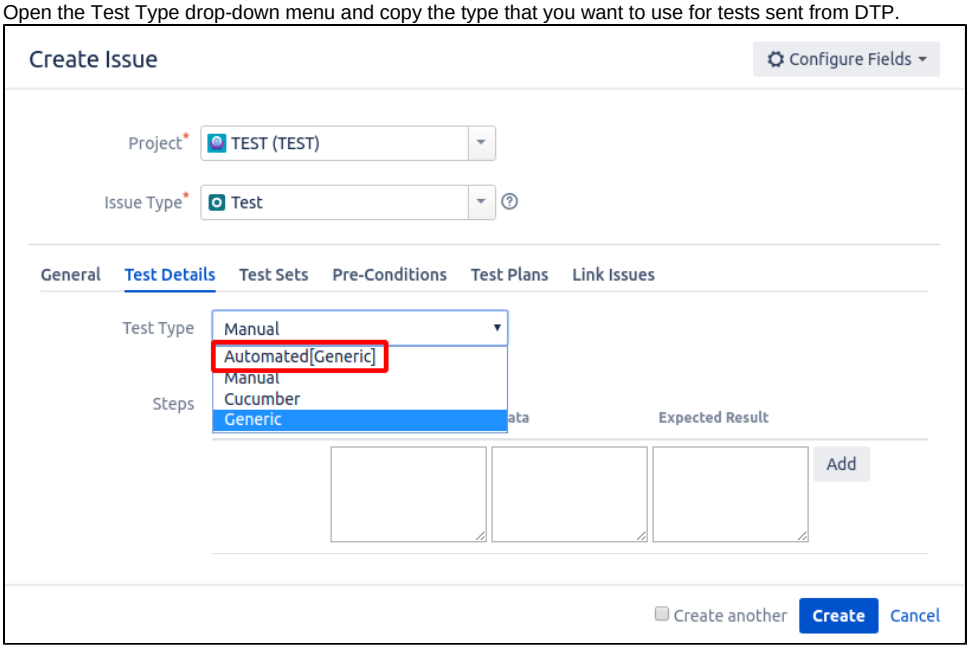
```
jira.testType.customfieldId=customfield_10800
```

To get the value for the `jira.testType.customfieldValue` setting:

1. Log into your Jira system and click **Create** in the Jira toolbar.



2. Choose **Test** (or equivalent) from the Issue Type drop-down menu and click **Test Details**.
3. Open the Test Type drop-down menu and copy the type that you want to use for tests sent from DTP.



4. Set the `jira.testType.customfieldValue` property to the test type value:

```
jira.testType.customfieldValue=Automated[Generic]
```

Save the ExternalAppsSettings.properties configuration file to apply the changes.

Getting Default Test and Test Execution Statuses

The `jira.testDone.transitiondId` and `jira.testExecutionDone.transitionId` settings are used to specify a custom status for tests and test executions sent to Jira/Xray from DTP. You can use the Jira REST API to get the status names to use in the ExternalAppsSettings.properties file so that the data sent from DTP matches the existing entities in Jira.

Send a request to the following Jira `transitions` endpoint and specify the Jira test ID to determine the correct value for `jira.testDone.transitiondId` setting:

```
curl -X GET "http(s)://<jira_host>/rest/api/2/issue/<testId>/transitions"
```

The ID, as well as the status name, will appear in the response:

```
{
    "id": "41",
    "name": "Done",
    "to": {
        "self": "https://parasoft.atlassian.net/rest/api/2/status/10002",
        "description": "",
        "iconUrl": "https://parasoft.atlassian.net/",
        "name": "Done",
        "id": "10002",
        "statusCategory": {
            "self": "https://parasoft.atlassian.net/rest/api/2/statuscategory/3",
            "id": 3,
            "key": "done",
            "colorName": "green",
            "name": "Done"
        }
    },
    "hasScreen": false,
    "isGlobal": true,
    "isInitial": false,
    "isConditional": false
}
```


Specifying the ID in the ExternalAppsSettings.properties file:

```
jira.testDone.transitiondId=41
```

Send a request to the following Jira `transitions` endpoint and specify the Jira test execution ID to determine the correct value fo ther `jira.testExecutionDone.transitiondId` setting:

```
curl -X GET "http(s)://<jira_host>/rest/api/2/issue/<testExecutionId>/transitions"
```

The ID, as well as the status name, will appear in the response:


```
"expand": "transitions",
"transitions": [
    {
        "id": "31",
        "name": "In Progress",
        "to": {
            "description": "This issue is being actively worked on at the mo
            "iconUrl": "https://jira-stage.parasoft.com/images/icons/statuse
            "id": "3",
            "name": "In Progress",
            "self": "https://jira-stage.parasoft.com/rest/api/2/status/3",
            "statusCategory": {
                "colorName": "yellow",
                "id": 4,
                "key": "indeterminate",
                "name": "In Progress",
                "self": "https://jira-stage.parasoft.com/rest/api/2/statusca
            }
        }
    },
```


Specifying the ID in the ExternalAppsSettings.properties file:

```
jira.testExecutionDone.transitiondId=31
```

Usage

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

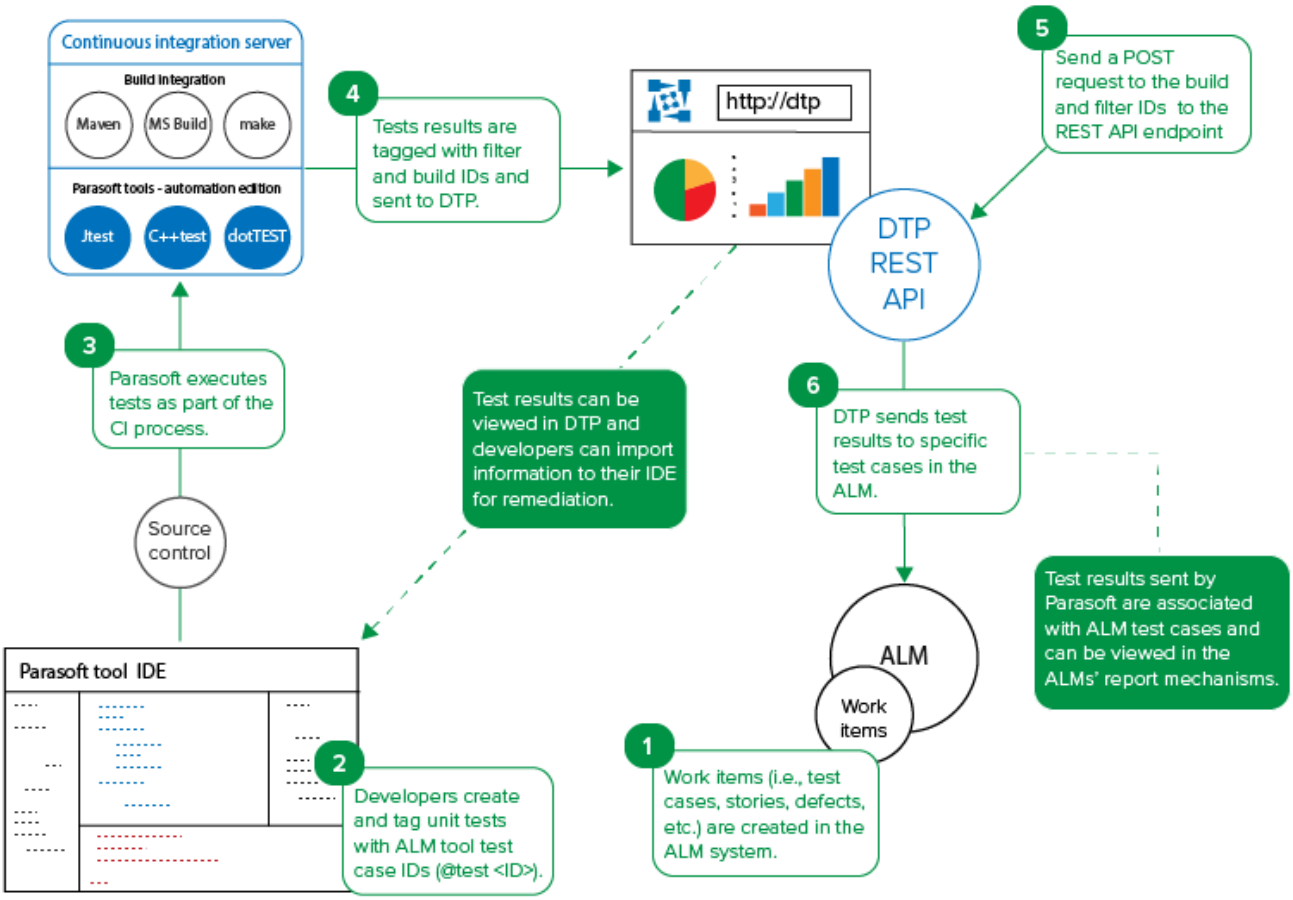
Manually Creating Bugs and Tasks in Jira

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

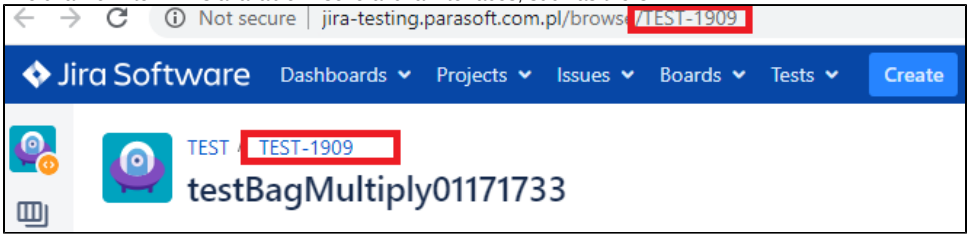
- See Creating an Issue in a Third-party System for instructions on how to manually create bugs and tasks in Jira from the Violations Explorer view.
- See Creating an Issue in a Third-party System for instructions on how to manually create bugs and tasks in Jira from the Test Explorer view.

Sending Test Data to Jira/Xray

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



1. Create tests and/or stories in Jira. The items will be associated with tests executed by Parasoft C/C++test, dotTEST, or Jtest.
2. In your test file, add the Jira test or story IDs using the `@test` or `@req` annotation. See the C/C++test, dotTEST, or Jtest documentation for details on how to add annotations.
    - Use the `@test <Jira Test ID>` annotation to associate tests with test executions in Jira.
    - Use the `@req <Jira Story ID>` annotation to associate tests with stories in Jira.
    - The Jira work item ID is available in several Jira interfaces, such as the URL:



3. Execute your tests as part of the CI process. You can also manually execute the tests from the IDE.
4. As part of the test execution, Parasoft test execution tools 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. 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 Jira when Data Collector receives the data from the Parasoft tool. You can also manually send a POST request to the DTP REST API endpoint in order to send results from the DTP database to Jira (see Manually Send a POST Request to the DTP REST API Endpoint).
6. DTP will locate the test results that match the filterId and buildId parameters and send the data to the items in Jira.
    - When DTP locates results with an @test <ID>, it will search for and update tests with a matching ID in Jira. No action will be taken if the IDs do not exist in Jira.
    - When DTP locates results with an @req <ID>, it will search for Jira stories with a matching ID and add test executions to the story. If the ID does not exist, new test executions will be added to Jira.
    - 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 Jira.

After DTP processes the report and sends results to Jira, 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"
     ]
}
```

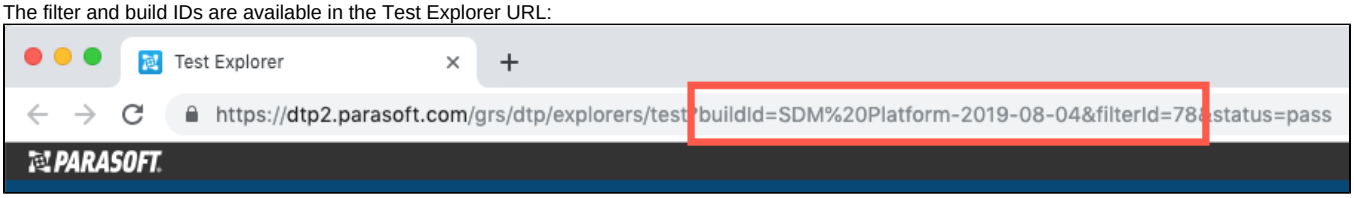
Manually Sending a POST Request to the DTP REST API Endpoint

The **Sending Test Data to External Application Flow** shipped with the Traceability Pack will automatically send data from DTP to Jira when new results are collected in DTP (see Deploying the Sending Test Data to External Application Flow). Alternatively, you can send a POST request to the DTP REST API endpoint to start this action.

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:



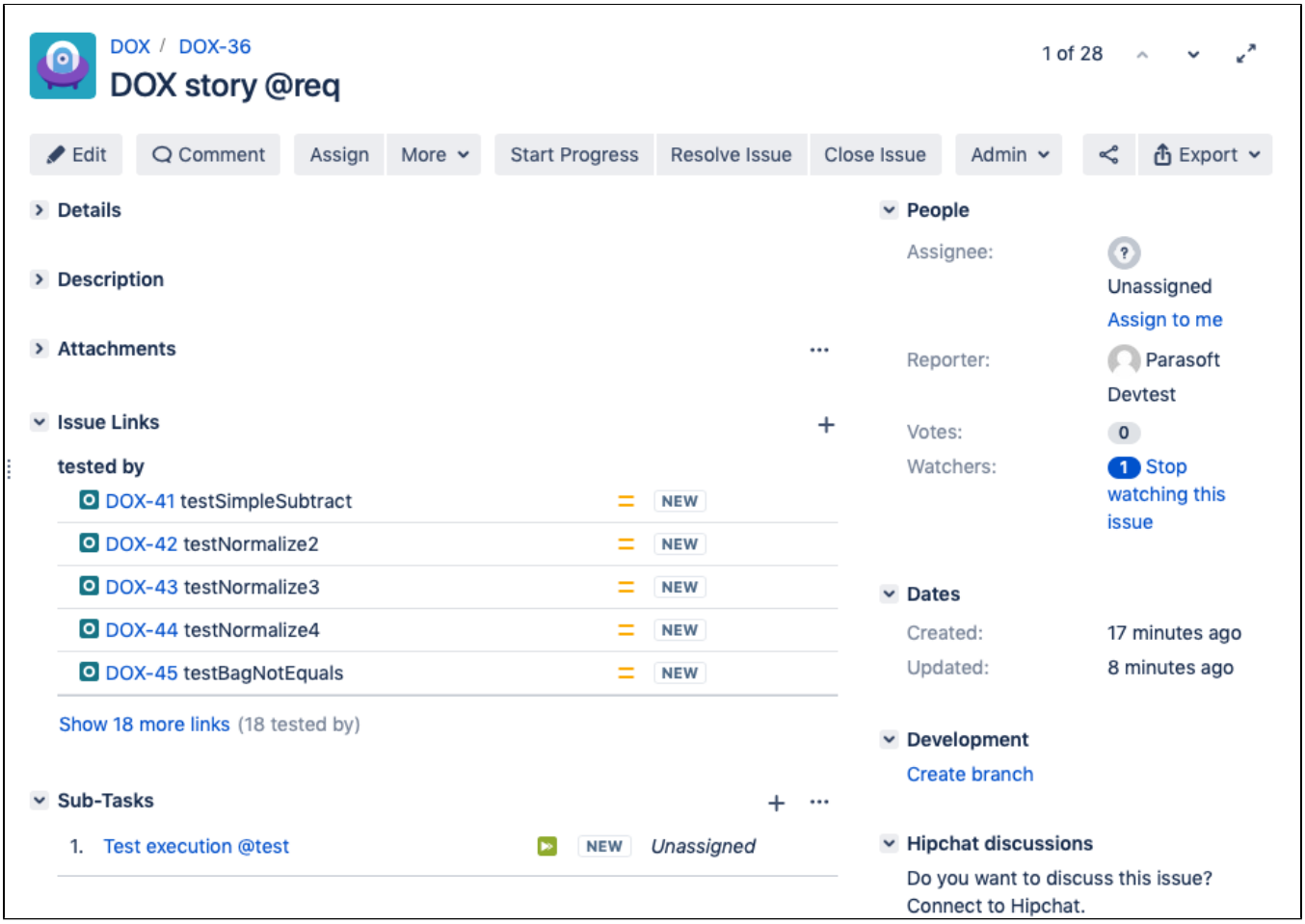
You can also specify a one or more values for Jira's fixVersions field:

```
curl -X POST "https://<host>:<port>/grs/api/v1.7/linkedApps/configurations/1/syncTestCases?
filterId=<filterID>&buildId=<buildID>&fixVersions=<version1>,<version2> "
```

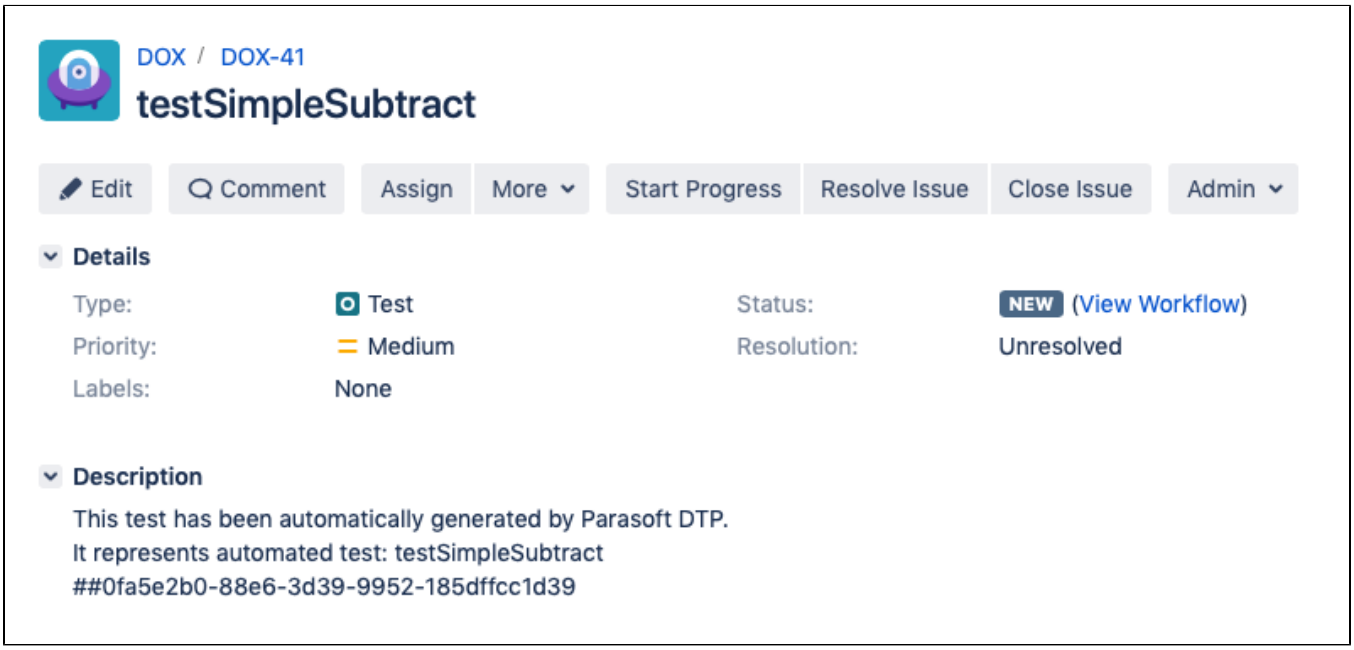
The `fixVersions` parameter is optional. The fixVersions field will be empty in Jira if the property is not included.

Viewing Results in Jira and Xray

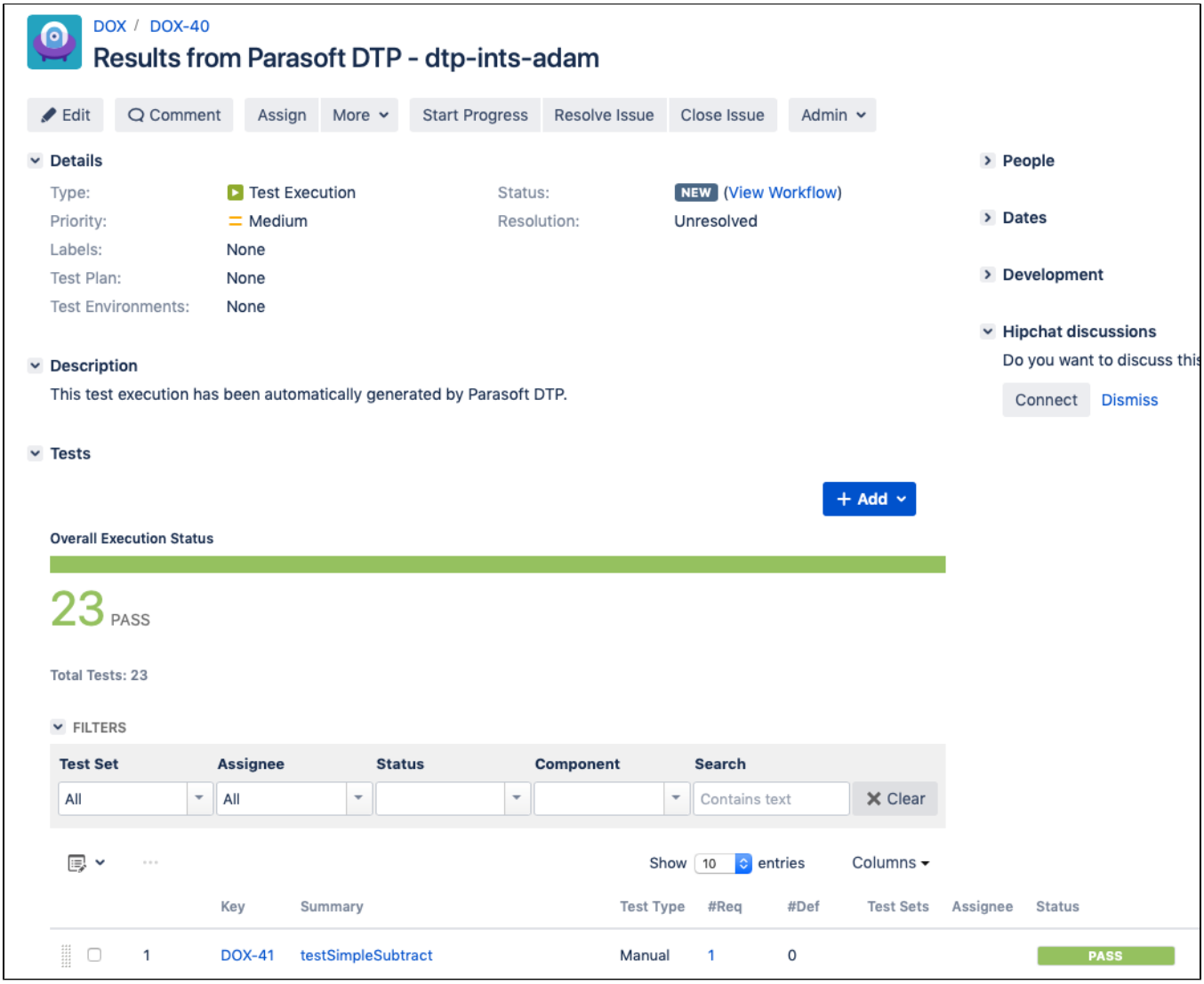
You will be able to view results in Jira after sending the test data. The following image shows a Jira story that contains several tests.



You can click on a test associated with the story to view additional details.



You can also drill into test executions associated with the test.

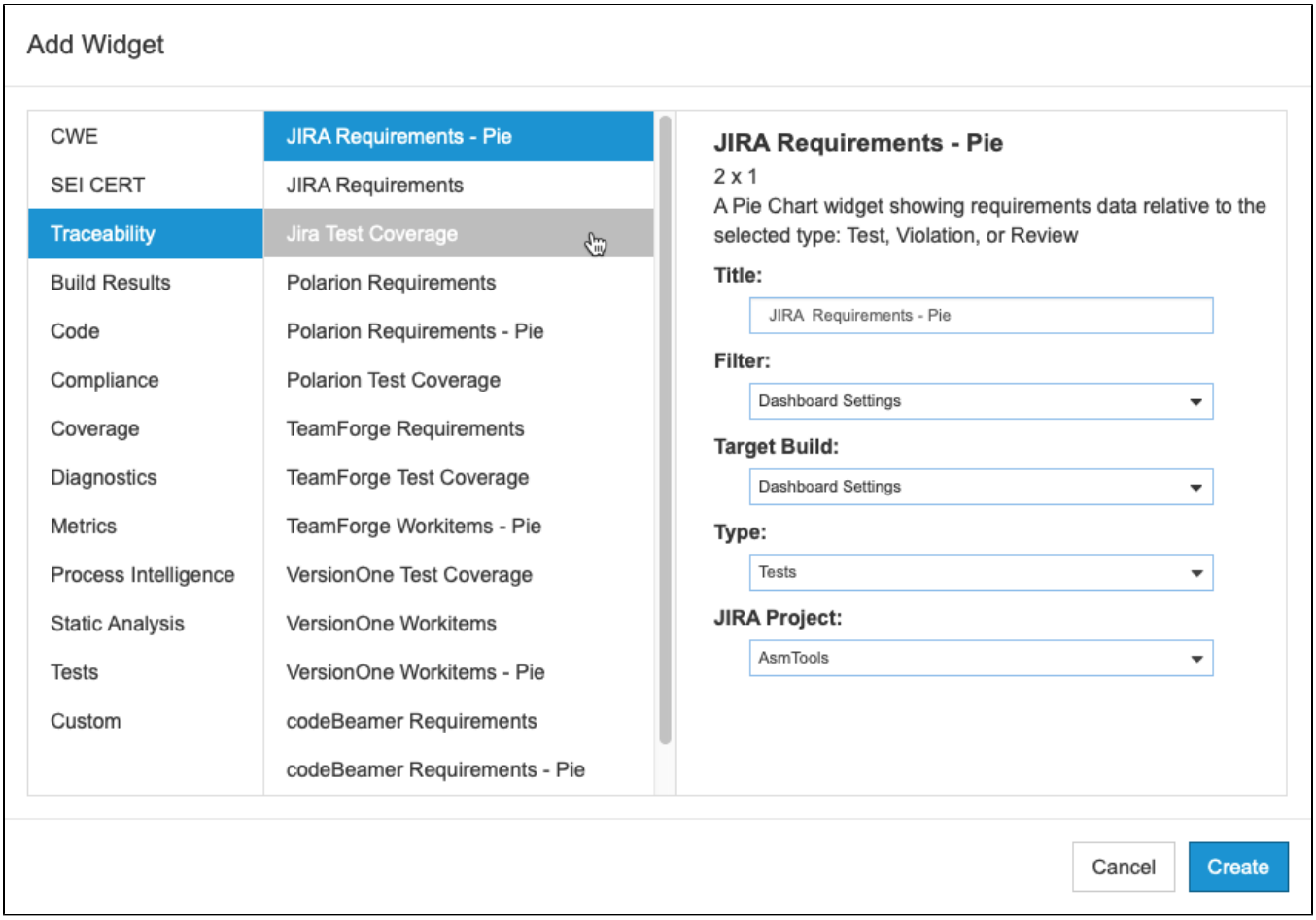


Viewing the Traceability Report

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.

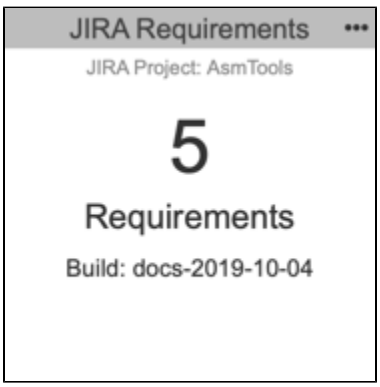


You can configure the following settings:

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 Jira project from the drop-down menu.

Requirements Widget

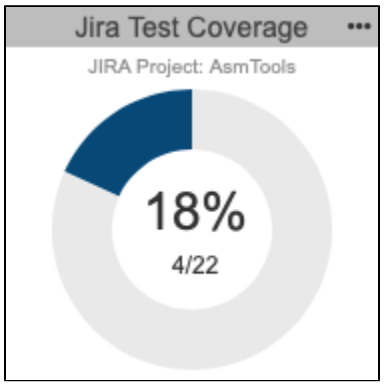
The widget shows the number of requirements from the specified Jira project.



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.

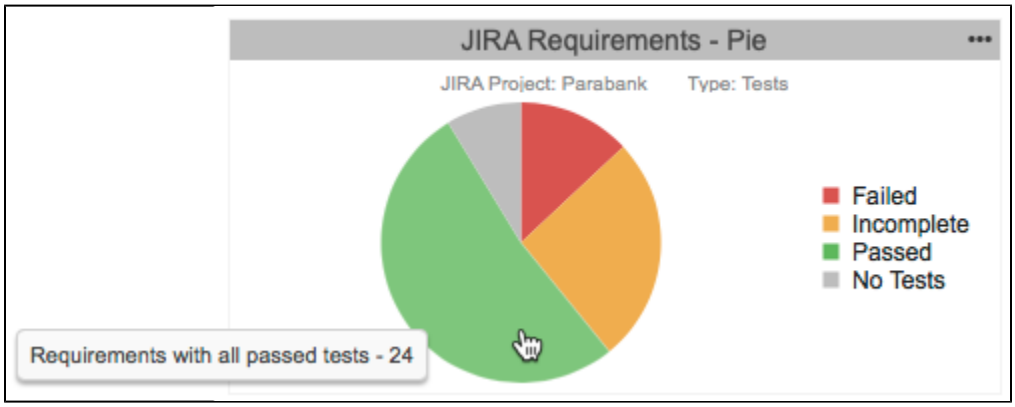
The colored-in segment represents the requirements covered by tests. Click on the segment to open the Requirement Traceability report filtered to the **With Tests** category.

Pie Widget

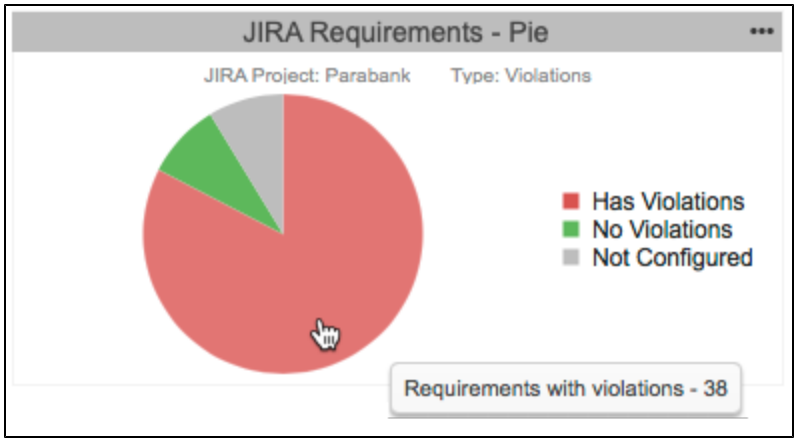
Unit testing, functional testing, static analysis, and peer reviews are common activities for verifying that requirements have been properly and thoroughly implemented. This widget shows the overall status of the project requirements in the context of those software quality activities. You can add a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

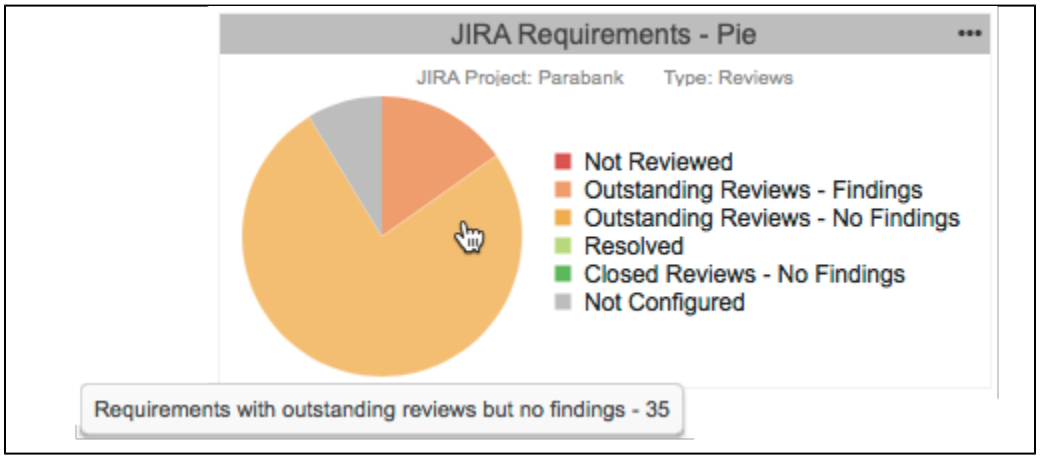
Requirements Implementation Status by Tests



Requirements Implementation Status by Violations

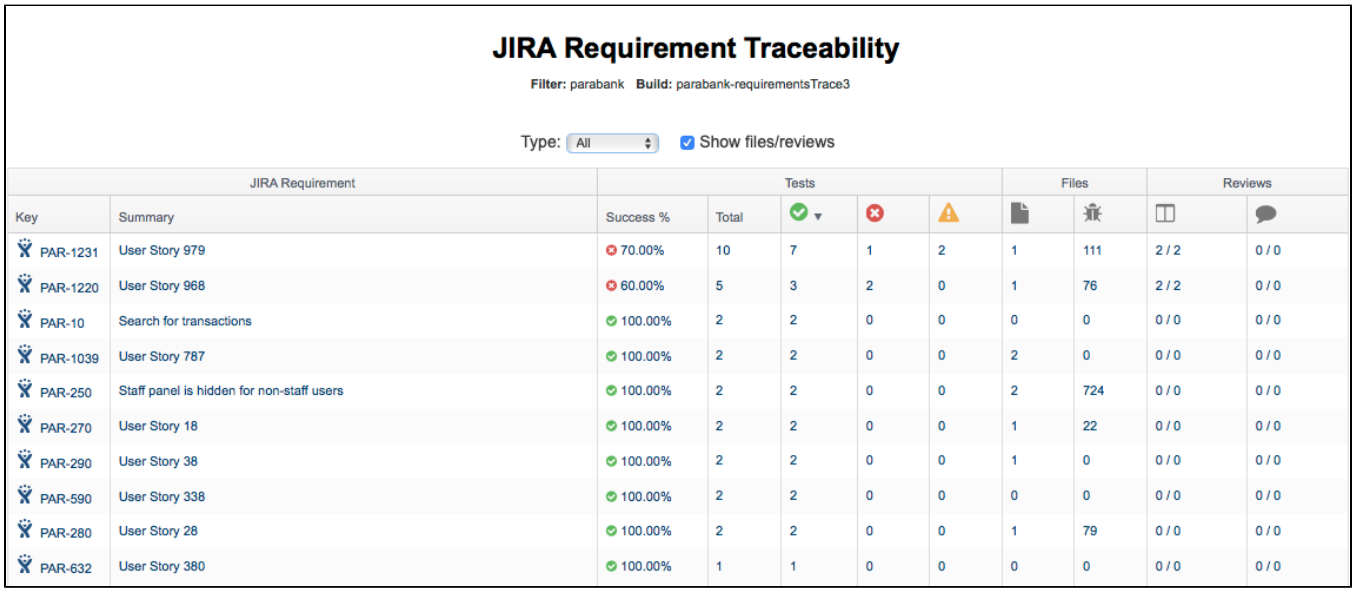


Requirements Implementation by Reviews



Understanding the Requirement Traceability Report

The report lists the JIRA requirements and data associated with them.



JIRA Requirement Traceability

Filter: parabank **Build:** parabank-requirementsTrace3

Type: All Show files/reviews

JIRA Requirement		Tests					Files		Reviews	
Key	Summary	Success %	Total							
PAR-1231	User Story 979	70.00%	10	7	1	2	1	111	2 / 2	0 / 0
PAR-1220	User Story 968	60.00%	5	3	2	0	1	76	2 / 2	0 / 0
PAR-10	Search for transactions	100.00%	2	2	0	0	0	0	0 / 0	0 / 0
PAR-1039	User Story 787	100.00%	2	2	0	0	2	0	0 / 0	0 / 0
PAR-250	Staff panel is hidden for non-staff users	100.00%	2	2	0	0	2	724	0 / 0	0 / 0
PAR-270	User Story 18	100.00%	2	2	0	0	1	22	0 / 0	0 / 0
PAR-290	User Story 38	100.00%	2	2	0	0	1	0	0 / 0	0 / 0
PAR-590	User Story 338	100.00%	2	2	0	0	0	0	0 / 0	0 / 0
PAR-280	User Story 28	100.00%	2	2	0	0	1	79	0 / 0	0 / 0
PAR-632	User Story 380	100.00%	1	1	0	0	0	0	0 / 0	0 / 0

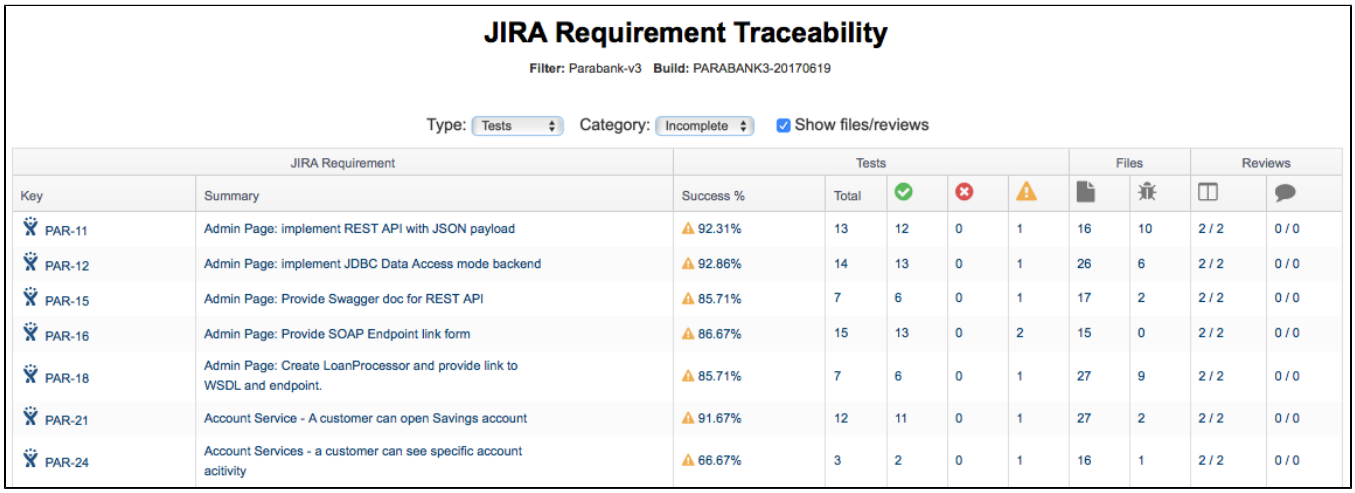
You can perform the following actions:

- 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 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 Details report for the requirement.

- 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 inthe Requirement Details report.

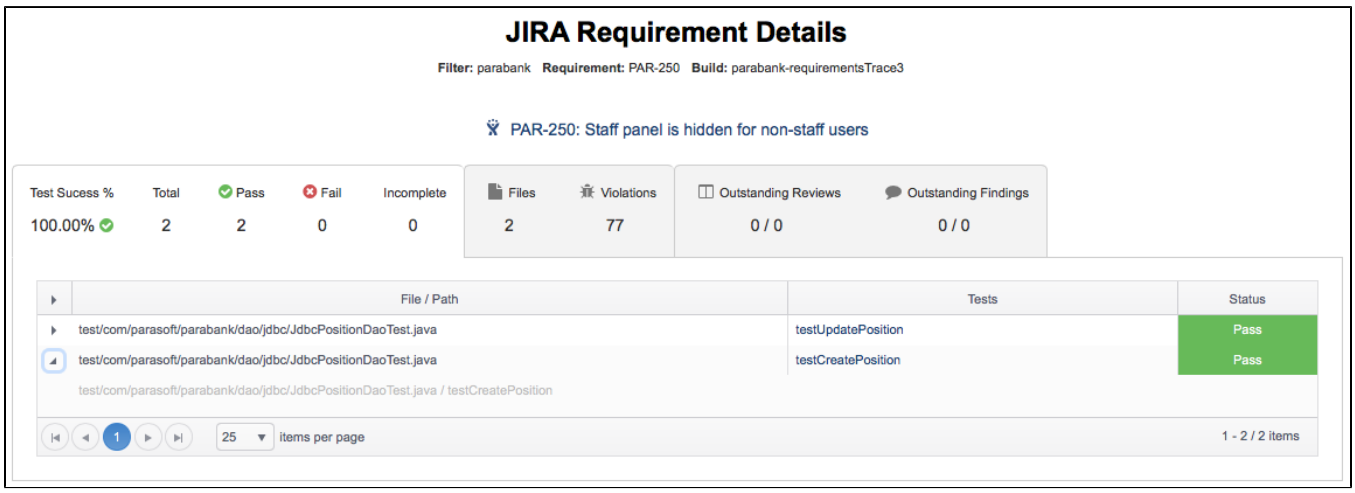
Requirement Traceability Report by Type

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.



Understanding the Requirement details Report

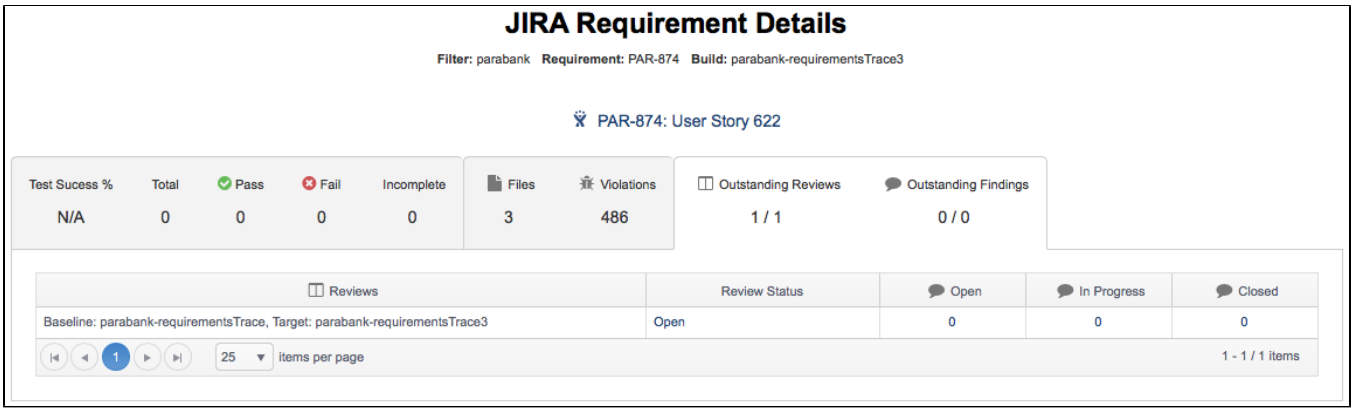
The Requirement Details report provides additional information about the files and tests associated with the Jira requirement. You can open this report by click on a requirement in the Requirement Traceability report.



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 violation 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.



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.