Parasoft Guide to Requirements Traceability

In this guide:

Introduction

Requirements describe functional aspects of an application, such as the effect of clicking a button in the UI, as well as non-functional characteristics, such as the application's resilience to cyber attacks. Requirements are commonly created and tracked in application lifecycle management (ALM) or requirement management systems (RMS). In some ALM/RMSs, "requirements" are just one type of "work item" for tracking the application under development.

Developers and QA engineers create unit tests and functional tests to verify and validate that the software meets the requirements. But tracing tests to the requirements that they are intended to verify is challenging, however, due to the complexity, speed, and scale of modern applications. Organizations that commit to tracing requirements to tests and vice versa often rely on Excel spreadsheets with manually-populated test execution results. In some industries, such as defense, automotive, and medical devices, any code that affects the safety and security of the application *must* be tested according to industry standards *and* traced to the requirements that define the safety and security levels.

Parasoft's traceability functionality solves these challenges by enabling you to automate test and requirements tracing as part of your test execution workflow. Parasoft DTP also enables you to quickly generate traceability reports that satisfy regulatory compliance requirements. This guide describes how to configure Parasoft products to achieve requirements traceability.

Audience

This guide is intended for people who perform the following roles:

- Developers who create and run unit tests against their code
- Test engineers who create and run functional and UI tests against the application
- Parasoft DTP administrators
- Managers responsible for collecting and defining requirements
- Managers responsible for providing traceability information during compliance audits

Supported Environments

Most of the functionality that enables traceability is performed by Parasoft DTP. The Traceability Pack for DTP must be installed to generate the traceability dashboard widgets and reports.
The following table describes which Parasoft tools work with the supported ALM/RMSs.

	codeBeamer	Jama Connect	Jira [1]	Polarion [2]	TeamForge	VersionOne
C/C++test Professional	X [3]	X	X [3]	X [3]	X	X
dotTEST	X	X	X	X	X	X
Jtest	X	X	X	X	X	X
Selenic	X	X	X	X	X	X
SOAtest	X [3]	X	X [3]	X [3]	X	X

[1] The Xray extension is required for full functionality.
[2] Additional integration files must be installed in DTP. Refer to the DTP documentation for details.
[3] The Requirements View, which is an additional interface for interacting with ALM/RMS requirements in the desktop, is only supported for these systems.

Workflow Overview

In this guide, we assume that your Parasoft test execution tool is configured to send results to DTP. The following procedure provides an overview of the traceability workflow:

1. Create requirements in your ALM/RMS.
2. Associate requirement IDs with tests using code-level annotations.
3. Configure DTP to connect to your ALM/RMS and deploy the Traceability Pack to your DTP environment. The Traceability Pack includes reports, widgets, and other artifacts that facilitate traceability with Parasoft.
4. Execute the tests.
5. When DTP collects the results, traceability widgets and reports will indicate test coverage for your requirements. Results are also reported to the ALM/CMS, providing backward and forward visibility. You can also create issues in the ALM/RMS for test failures. Test and requirements data can be sent automatically or manually.
6. Developers and test engineers can download the tasks associated with the test results directly into their IDEs for remediation.

Creating Requirements and Work Items

Refer to the documentation for your ALM/RMS for instructions on how to create artifacts that you want to link to your tests.

Custom Requirements Configurations

Parasoft's out-of-the-box integrations are configured to work with default or commonly-used fields and work item types, but you can configure the *ExternalSystemSettings.properties* configuration file located in the <DTP_DATA_DIR>/conf directory to change the default behavior of the integration. This enables you to map tests and issues created in the ALM/RMS from the DTP interface to custom work items. Refer to the documentation for each integration for additional details.

- Integrating with CodeBeamer ALM
- Integrating with Jama Connect
- Integrating with Jira
- Integrating with Polarion ALM
- Integrating with TeamForge
- Integrating with VersionOne

This guide covers the default configuration.

Associating Tests and Code with Requirements

Create a link between your tests and the requirements stored in the ALM/RMS. You can create the association by adding special annotations to your unit tests and specifying the requirement or work item IDs that you want to link. You can use the dedicated GUI in SOAtest to create the link between requirements/work items and functional tests. The dedicated GUI for creating test associations is also available in C/C++test Professional.

Several tags are available for annotating tests, but on the `req` and `test` tags are supported for requirements and test traceability.

Jtest and dotTEST

Before you can use the annotations, you must enable the report.associations property in the *<tool>.properties* configuration file:

```
report.associations=true
```

Open your test files and create the associations using the following syntax:

```
@<tag> <REQUIREMENT_ID>
```

In the following example, the test class is associated with a requirement in the ALM/RMS with the ID PROJECT-123 and a test within the class is associated with work item ID PROJECT-456:

```
/**
 * @req PROJECT-123
 */
public class Test {

    testSomething1()
    {
     ...
    }

    /**
     * @test PROJECT-456
     */
    testSomething2()
    {
      ...
    }
}
```

C/C++test Professional

You can use the same association mechanism and syntax as Jtest and dotTEST to manually associate test suites with requirements. You can also use the Requirements View to automatically import the requirements from supported ALM/RMSs into the IDE so that you can use the C/C++test GUI to associate tests (see Supported Environments). Refer to your C/C+test documentation for complete information about the Requirements View.

Importing Requirements

1. Choose **Parasoft > Show View > Requirements** in the IDE menu to open the Requirements view.
2. Choose **Import** in the view's menu and choose **DTP...** or **Local File...** to open an import dialog.
3. Choose one of the following options:
    - If you import from a file, specify the path the ReqIF exported from your RMS and select the specification you want to import from that file.
    - If you import from DTP, choose the specification you want to import from your project in DTP. DTP must be connected to your ALM/RMS in order to use this option (see Connecting DTP to the ALM/RMS)
4. Click **OK** and the imported data will appear in the Requirements View.

Reviewing Requirements

Depending on your RMS, a requirement may include one or more sub-modules called "test definitions". All work items (requirements and their test definitions, if available) are arranged as nodes in a tree, which can be collapsed or expanded.

Associating Test Definitions with Requirements

Right-click a requirement in the Requirements View and choose **Copy ID** to copy its ID to your clipboard. This helps you correlate a requirement or test definition with test cases by enabling you to paste the ID in your code to create an annotation as described in Jtest and dotTEST.

SOAtest

You can use the Requirements View to automatically import the requirements from supported ALM/RMSs into the IDE (see Supported Environments). You can use the Requirements View to import requirements from a ReqIf file. Alternatively, the Requirements and Notes tab of SOAtest's test suite configuration panel enables you to annotate tests using the same syntax as described in Jtest and dotTEST.

Requirements View

- Import requirements into the Requirements View as described in the C/C++test Professional section (see Importing Requirements).
- After loading the requirements, choose one of the following methods for associating requirements with tests:
    - Drag requirements onto test or test suite in the Test Case Explorer
    - Drag the test or test suite from the Test Case Explorer onto a requirement.
    - Right-click a requirement in the Requirements View and choose **Copy ID** so that you can manually add the requirement ID as described in Requirements and Notes Tab.

Requirements and Notes Tab

1. Double-click on a test suite folder and click the **Requirements and Notes** tab.
2. Choose a test or nested test suite and click **Add**. Adding an association at the test suite level associates all tests in the suite with the same requirement ID. This is in addition to any IDs you want to associate individually.
3. Choose either the **@req** or **@test** annotation from the Type menu. SOAtest includes other annotation types, but by default only the @req and @test types are supported in the traceability workflow described in this guide.
4. Specify the requirement ID and optional URL.

5. Click **OK**.

Selenic

The Parasoft Annotations dependency must be added to your project so that you can add annotations to your Selenium tests and associate them with work items (requirements) stored in your ALM/RMS. If you used Selenic's project creation functionality, then the dependency will already be added. For existing projects, add the following dependency to your project pom.xml file:

```
<dependency>
        <groupId>com.parasoft</groupId>
        <artifactId>parasoft-annotations</artifactId>
        <scope>test</scope>
        <version>1.0</version>
</dependency>
```

For full details about the dependency, refer to the Parasoft Annotations project on GitHub, which includes the JavaDoc API documentation.

Adding Annotations

Use the `@WorkItem` annotation in your test methods and classes using the following syntax to correlate the test with work items in your ALM.

```
@WorkItem(type=<type>, id="<ID>", url="<URL>")
```

Annotations include the ID of the work item, type of work item (requirements, in this guide), and a URL to the work item in the ALM (optional).

In the following example, the function is associated with a requirement ID REQ-123:

```
@WorkItem(type=REQ, id="REQ-123, url="https://server.myALM.com/")
public void myFunction() {
. . .
}
```

Configuring DTP

A connection to the ALM/RMS must be configured in DTP in order to send and retrieve data from the system. To generate traceability reports and other visualizations, the Traceability Pack must also be installed in DTP and the visualization tools deployed to the DTP environment. In this guide, we assume that you have correctly configured filters in DTP to match the incoming data from your Parasoft tools.

Connecting DTP to the ALM/RMS

1. Choose **Report Center Settings** from the settings (gear icon) drop-down menu.
2. Choose **External System** and choose the type of ALM/RMS from the System type drop-down menu.
3. Enable the **Enabled** option.
4. Enter a name for your instance of the ALM/RMS in the Name field. The name is required, but it does not affect the connection settings or render in any other interfaces.
5. Enter the URL for the ALM/RMS 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 ALM/RMS system are created. This URL should include additional paths that may be necessary to access the system in a browser.
7. Enter login credentials in the Username and Password/API token fields. The login must have sufficient privileges to create issues in the ALM/RMS projects specified in the Project Associations section. If you are connecting to a cloud-hosted Jira system, enter the API token into the field.
8. (Optional) If you are connecting to a Jira system, you can specify a JQL query to filter the Jira requirements presented in Parasoft IDE plug-ins and Jira traceability reports. The JQL query applies to requirements in Jira projects that have been associated with DTP projects (see Associating Parasoft Projects) and to requirements that have been customized with the `jira.issueType.requirement` setting (see Creating Requirements and Work Items).
9. If you are connecting to the cloud-based Xray extension for Jira, enable the Xray (Cloud) enabled option. DTP will connect to hosted instances of Xray using the main Jira connection.
10. Click **Test Connection** to verify your settings and click **Save.**

Associating Parasoft Projects

Configuring the project association enables you to manually create issues from the DTP Violations or Test Explorer views that are linked to the ALM/RMS project. It also enables DTP to send data to the correct project when syncing test results (see Sending Results to ALM/RMS).

You can associate multiple projects in DTP with an ALM/RMS project, but you cannot associate the same DTP project with more than one external 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 project in the External Project field and click **Create** to save the association.

You can remove an association by clicking the trash icon. If an association is deleted and recreated later, existing links between violations and external issues will be reactivated.
You can reconfigure an existing association between DTP and external 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 project in the External Project field.
2. Click **Save**.

Installing the Traceability Pack

The Parasoft Traceability Pack is a set of artifacts for your DTP infrastructure that help you establish and demonstrate traceability between tests, code analysis, and peer reviews.

- Log into DTP and choose **Extension Designer** from the settings menu (gear icon).
- Click the **Configuration** tab to open Artifact Manager.
- Click **Upload Artifact** and browse for the *traceability-pack-<version>.zip* archive.
- Click **Install** to complete the installation.

The Traceability Pack artifacts will be uploaded to DTP, but they will need to be deployed before you can use them.

Deploying Traceability Widgets and Reports

After installing the pack, you will need to deploy the artifacts to your DTP environment.

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.
3. If you are adding the artifact to an existing service, add a new Flow tab 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.

Deploying the Sending Test Data to External System Flow

This artifact automatically sends test data to your ALM/RMS when DTP Data Collector retrieves test results from a Parasoft tool. If you prefer to manually trigger this process or to automate it as part of your CI/CD pipeline, you can call the REST API endpoint as described in Sending Results to ALM/RMS.

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.
3. If you are adding the artifact to an existing service, add a new Flow tab 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**.

Including Static Analysis and Code Reviews in the Traceability Report

If you want to see static analysis and code review data in the traceability reports generated by DTP, you can also link analyzed source code files with requirements/work items. Doing so will add static analysis and code review tabs to the traceability details report, providing additional software quality or security information.

The following workflow describes how to enable requirements traceability for a specific build ID. You must repeat the steps for each additional build ID you want to include the traceability report.

1. Create a CSV file that maps the source code files for your project to the requirement IDs, e.g.:

File	Associated Requirement ID
Project-A/src/foo/goo.java	REQ-A
Project-A/src/foo2/goo.c	REQ-A, REQ-B, REQ-C

2. Add the file-to-requirement association information to the DTP database. You can either check the source code files into source control and execute the tests (see Test Execution) or call the `artifactsTypes` DTP REST API endpoint. Refer to the DTP API documentation for usage. You can access the API documentation from the DTP help link. In both cases, the build ID and filter ID must be specified.
3. Scan the CSV file by running the fileReqAssoc.groovy script shipped with DTP in the <DTP_INSTALL>/grs/extras/traceability directory. Specify the CSV file, build ID, DTP host and protocol, and DTP login credentials:

```
groovy fileReqAssoc.groovy -csv <CSV_FILE_NAME> -build <BUILD_ID> -dtp <DTP_HOST_INC_PROTOCOL> -user
<DTP_USERNAME> -password <DTP_PASSWORD>
```

a. This is an example script intended to demonstrate a sequence of API calls that should be performed to associate files with artifacts in DTP. You can use this example script as a starting point for implementing a more advanced solution.
b. The script is written in Groovy, but the directory also contains a JAR file that you can run with the same arguments.
c. The script scans the CSV file prepared in step #2 and feeds the DTP database with the information about requirements to files mapping. See the README.txt file for details how to run the script.

After the script finishes, the traceability data for the specific filter ID and build ID will be stored in the database.

Test Execution

Execute the tests that you associated with the requirements (see Associating Tests and Code with Requirements). Refer to the documentation for your tool for details on how to execute tests. When the results are reported to DTP, you can use the special dashboard widgets and reports to monitor requirements coverage.

Sending Results to ALM/RMS

If you deployed the **Sending Test Data to External System** flow to DTP, then this step will be performed automatically when DTP receives the test results. Otherwise, you can manually call the REST API endpoint in DTP to send test data to your ALM/RMS.

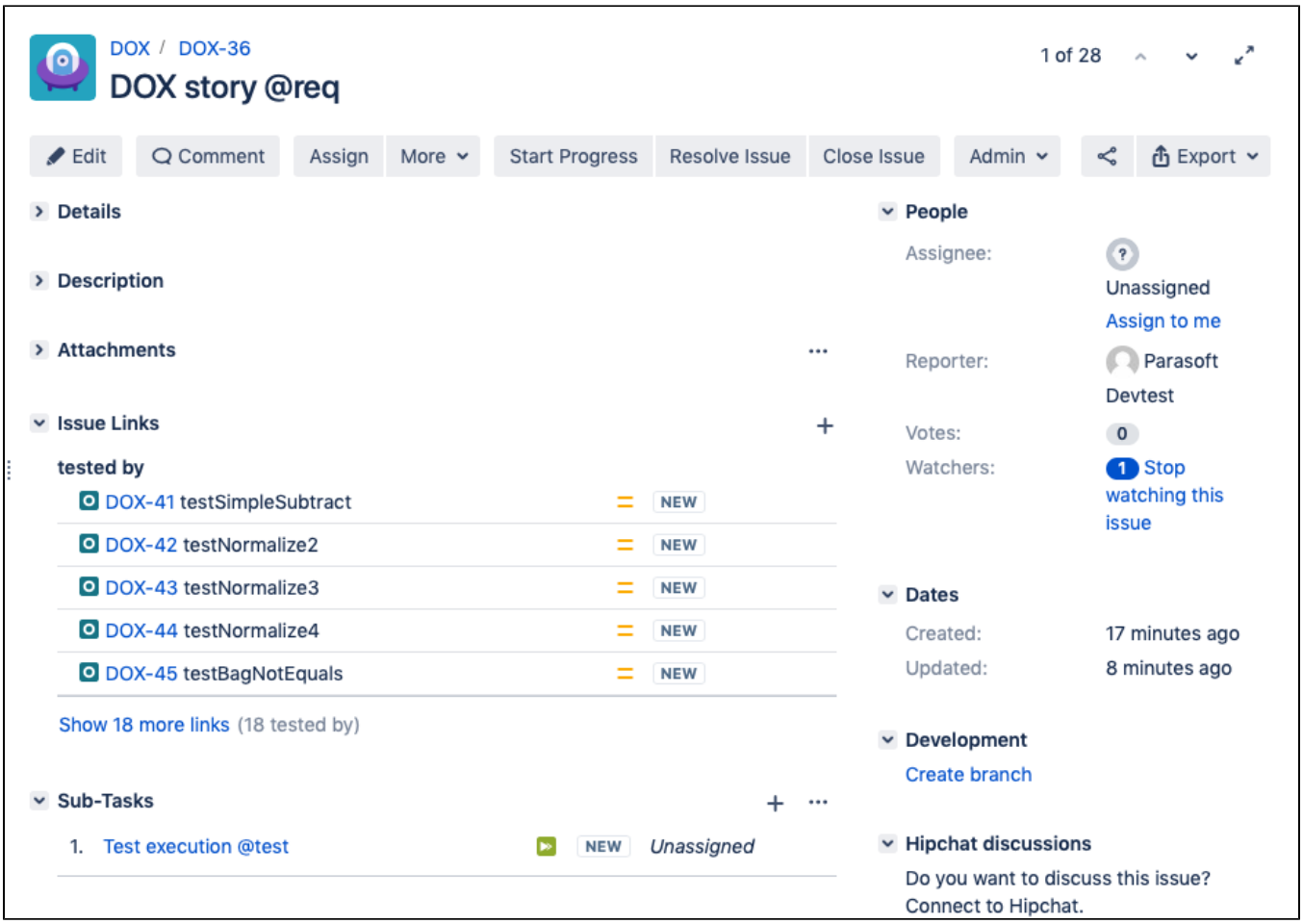
Resource	/syncTestCases
URL	<PROTOCOL>://<DTP_HOST>:<DTP_PORT>/grs/api/v1.7/linkedApps/configurations/1/syncTestCases
Method	POST
Parameters	• `filterId` (required) – You can get the filter ID value from the dashboard URL • `buildId` (required) • `fixVersions` (optional; Jira only)

Example Request

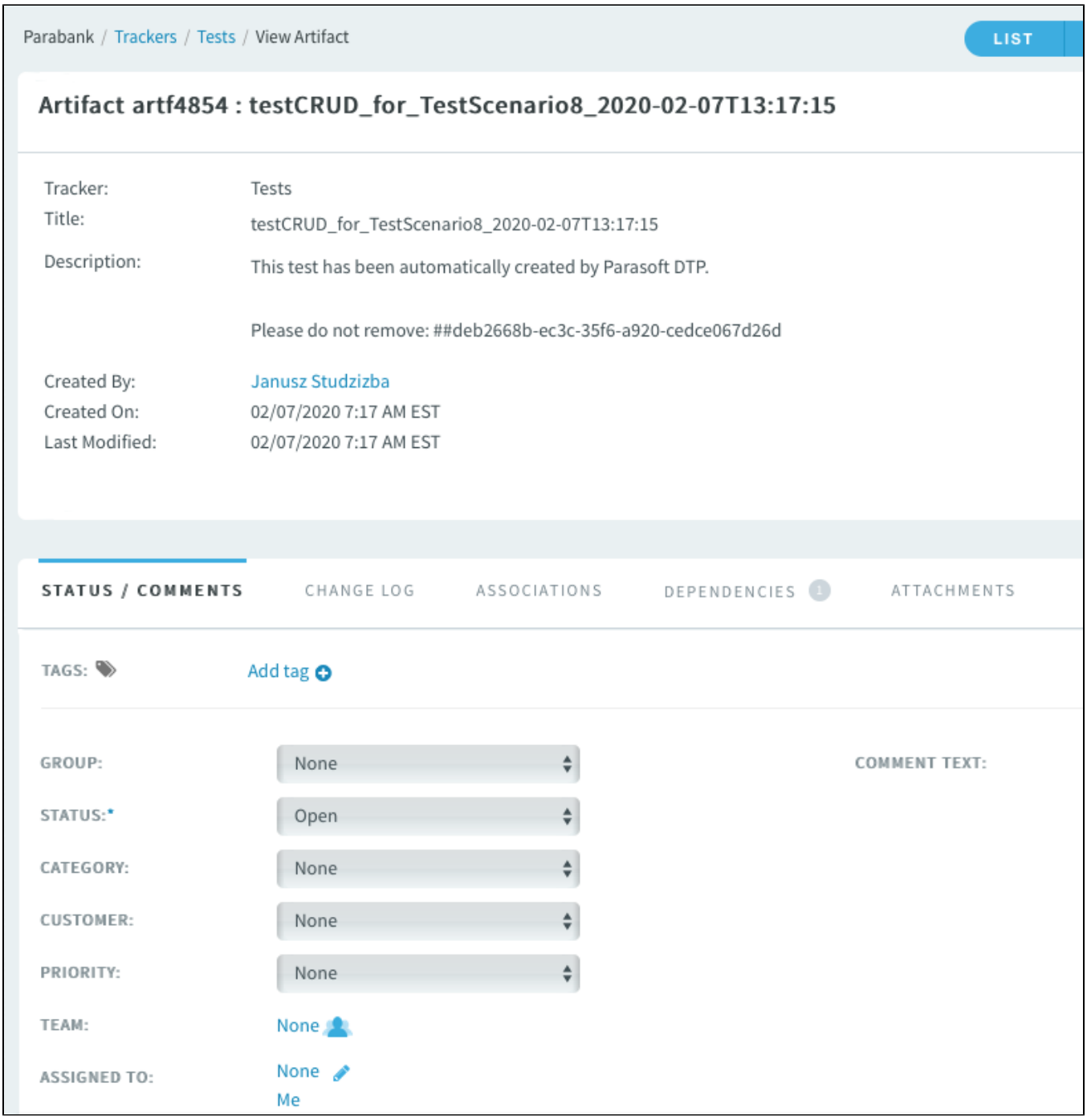
```
curl -X POST -u <username>:<password> "http://<host>:<port>/grs/api/v1.7/linkedApps/configurations/1
/syncTestCases?filterId=<filterID>&buildId=<buildID>"
```

Viewing Results in the ALM/RMS

You will be able to view results in your ALM/RMS after sending the test data. Refer to the documentation for details about viewing work items in your system. The following example shows a Jira story that contains several tests:



In most systems, you will be able to click on a test and drill-down to additional test execution details the include build and authorship information, as well as links back to the test or violation in DTP. The following example shows test details in TeamForge:



Viewing Traceability Information in DTP

Add the traceability widgets to your dashboard, which provide requirements coverage information, as well as link to the traceability report. The following widgets are available:

- Requirements: Shows the number of requirements from the specified project.
- Test Coverage: Shows the percentage of requirements covered by tests against all requirements in the project.
- Pie: 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.

Clicking on any widget opens the traceability report (see Traceability Report).

Adding and Configuring the Widgets

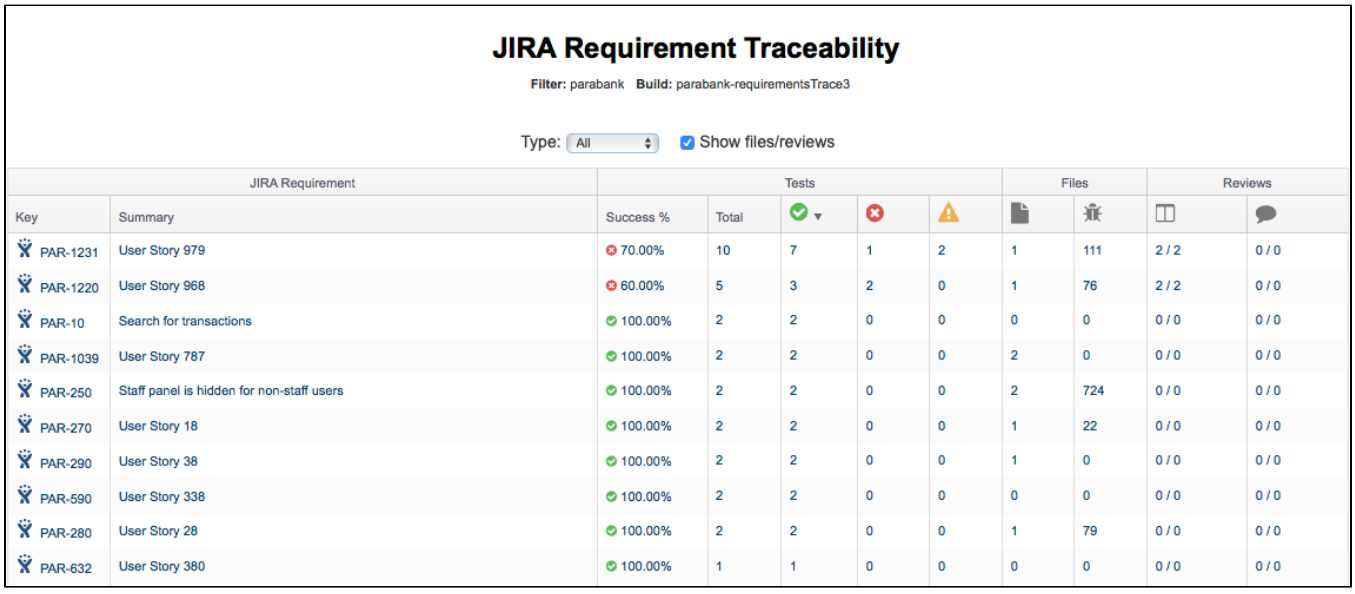
1. Open your DTP dashboard and click the Add Widget button.
2. Choose a widget from the Traceability category and configure the following settings.
    - Title: You can provide a more meaningful title or leave the default.
    - Filter: If you segmented project data into filters, specify the filter containing the test data you want to track. You can also leave the default setting and specify the filter for all widgets in the dashboard. Refer to the DTP documentation for information about filters and configuring dashboard settings.
    - Target Build: Specify the build ID containing the test data you want to track. You can also leave the default setting and specify the build ID for all widgets in the dashboard. Refer to the DTP documentation for information about build IDs and configuring dashboard settings.
    - Type (Pie widget only): Choose Tests, Violations, or Reviews from the drop-down menu. The widget will render a pie chart for the specified type of data. To render Violations and Reviews, the test files must have been associated with the requirements as described in I ncluding Static Analysis and Code Reviews.
    - <ALM/RMS> Project: Specify the project in your ALM/RMS containing the requirements you want to track. If you are unable to specify a project, verify that the connection settings are correct and that the credentials you provided have sufficient permissions to access the project (see Connecting DTP to the ALM/RMS)
    - Jama Set of Requirements (Jama only): Choose a set of items (e.g., set of requirements) from the selected project. You can choose any set of items in the project except sets of test cases and sets of defects.
    - JQL Query (Jira only): Specify a query to define which work items are presented in the widget. If left blank, the JQL specified in the integration settings will be applied (see Connecting DTP to the ALM/RMS). Queries defined in this field override the JQL configured in the integration settings.

Traceability Report

The traceability report provides an overview of your project's requirements coverage, as well as serves as the requirements traceability matrix (RTM) for auditing purposes. Click on any traceability widget to open the report. Some widgets can be configured to filter the data. As a result, clicking on the widget will open a pre-filtered report matching the data presented in the widget. Refer to the Parasoft DTP documentation for your ALM/RMS integration for additional information about viewing the traceability reports.

- Integrating with CodeBeamer ALM
- Integrating with Jama Connect
- Integrating with Jira
- Integrating with Polarion ALM
- Integrating with TeamForge
- Integrating with VersionOne

The following example shows the main requirement traceability report in an instance of DTP connected to Jira:



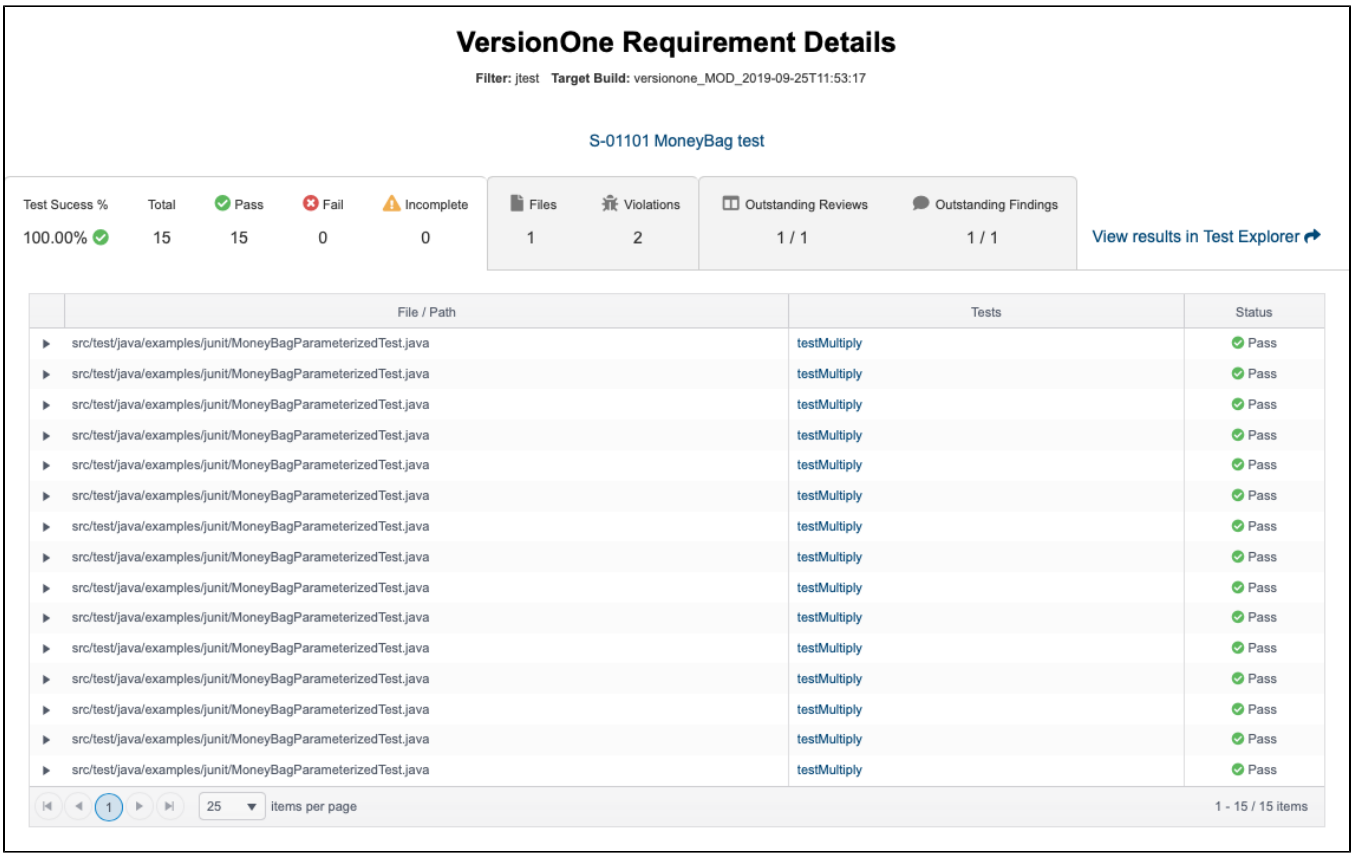
JIRA Requirement Traceability

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

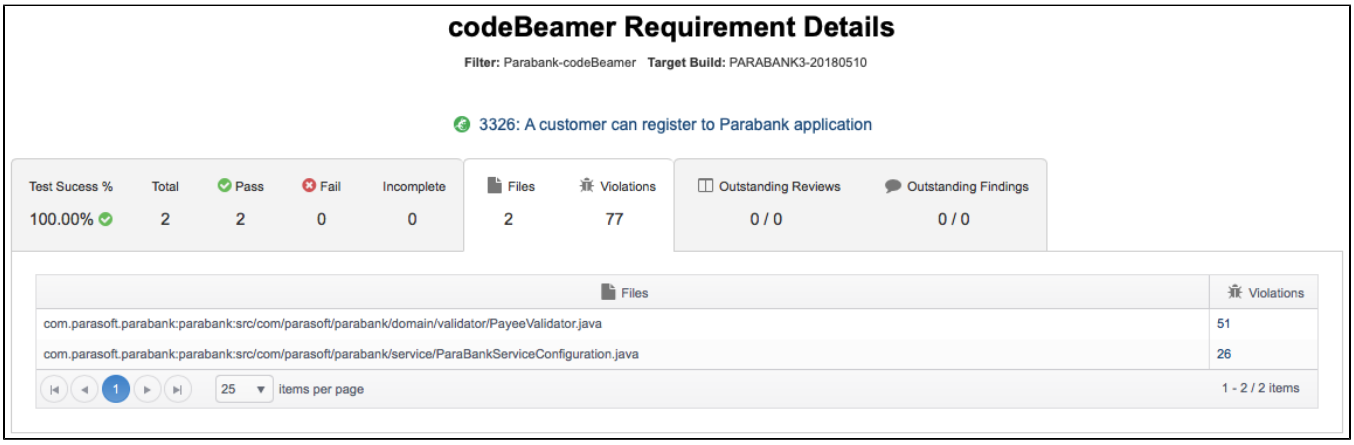
Type: All Show files/reviews

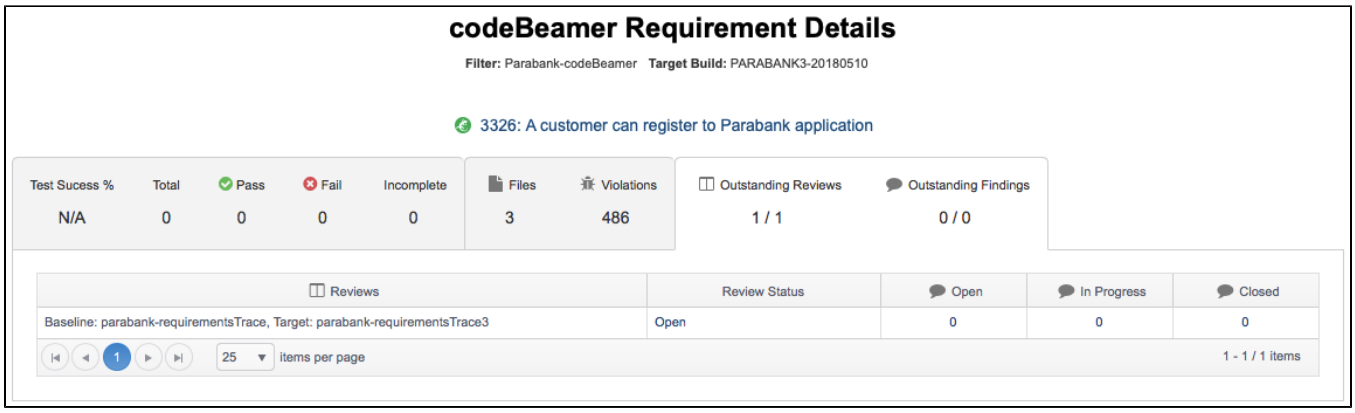
JIRA Requirement		Tests					Files		Reviews	
Key	Summary	Success %	Total	✓	✗	⚠				
PAR-1231	User Story 979	70.00%	10	7	1	2	1	111	2 / 2	0 / 0
PAR-1220	User Story 968	60.00%	5	3	2	0	1	76	2 / 2	0 / 0
PAR-10	Search for transactions	100.00%	2	2	0	0	0	0	0 / 0	0 / 0
PAR-1039	User Story 787	100.00%	2	2	0	0	2	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 click on a link in the Key column to drill down to the details report for individual requirements. The following example shows a details report for a requirement stored in VersionOne:



The details report shows the results of the tests that were executed to verify the specific work item. If you associated files with requirements/work items (see Including Static Analysis and Code Reviews), then the detail report will include tabs for static analysis and code reviews. The following examples show static analysis and reviews as part of a codeBeamer traceability details report.





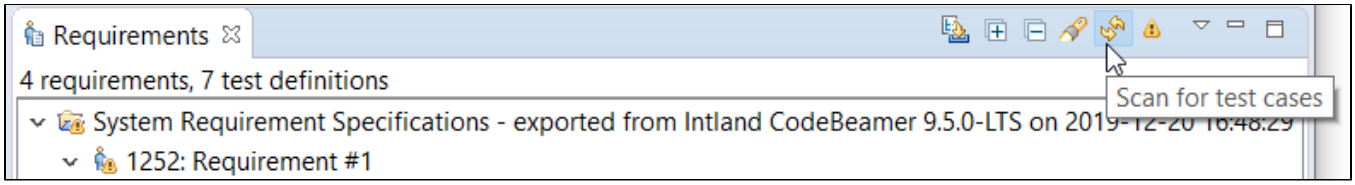
Covering Requirement Traceability Gaps

The Requirements View, which is available for the C/C++test Professional and SOAtest desktops, enables you to import requirements/work items stored in the ALM/RMS you're your IDE so that you can readily associate the requirements with tests in your workspace. The tool actually imports the data from Parasoft DTP, which with pulls the information from the ALM/RMS. The Requirements View is not currently available in Parasoft dotTEST, Jtest, and Selenic.
Refer to the following sections for instructions on importing requirements for your tool:

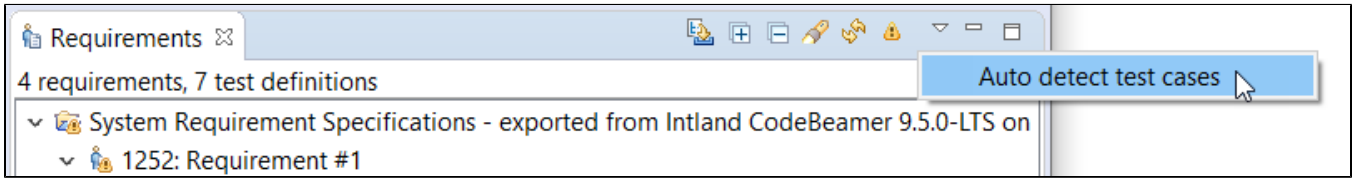
- Importing Requirements for C/C++test.
- Requirements View for SOAtest.

Scanning Requirements

After importing requirements, click the refresh icon on the Requirements view toolbar to scan for test cases as you work on your project.



You can also enable the **Auto detect test cases** option from the Requirements view toolbar menu to enable the automatic detection mode. As a result, the tool will automatically search for correlations as you make updates.



When the scanning process completes, the detected test cases are matched with the corresponding work item in the Requirements view. You can right-click a test case and choose one of the following options:

- Run the test with your choice of test configuration.
- Click **Remove Association**.
- Click **Open** to view the test case in Test Case Editor. This allows to review and customize its content. Alternatively, you can double-click a test case to open it in Test Case Editor.
- Click Show in Test Case Explorer to show the test case in Test Case Explorer.

Covering the Requirements

Follow the procedures described in the following sections to cover the requirements with any new tests you create for the project:

- Associating Test Definitions with Requirements for C/C++test
- Requirements View for SOAtest