Integrating with VersionOne

In this section:

Introduction

VersionOne is a popular browser-based platform for managing and tracking requirements, defects, and other work items. Parasoft DTP integrates with VersionOne, providing the following functionality:

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

Requirements

- 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 +
- To send test and analysis data to VersionOne, you should have already created work items in VersionOne. Parasoft can associate tests with the following types VersionOne work items:
  - Story
  - Test

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 VersionOne.

Connecting DTP to VersionOne Server

1. Choose **Report Center Settings** from the settings (gear icon) drop-down menu.
2. Choose **External Application** from the Administration sidebar and choose **VersionOne** from the Application Type drop-down menu.
3. Enable the **Enabled** option.
4. Enter a name for your instance of VersionOne in the Name field. The name is required but does not affect the connection settings or render in any other interfaces.
5. Enter the VersionOne URL in the Application URL field.
6. The Display URL field is rendered in DTP interfaces when links to your VersionOne system are created.
7. Enter login credentials in the Username and Password/API Tokens fields. The login must have sufficient privileges to create issues in the VersionOne projects specified in the Project Associations section.
8. Click **Test Connection** to verify your settings and click **Save**.

Associating Parasoft Projects with VersionOne Projects

Create links between Parasoft and VersionOne projects so that defects created in the Violations or Test Explorer view are created in the correct project in VersionOne. 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 VersionOne project in the External Project field and click **Create** to save the association.

Click the trash icon to remove a project association. Removing the project association does not remove links. If a new association is created, existing links between violations and VersionOne issues will be reactivated.

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

Enabling the Traceability Report

You can configure DTP to generate widgets and reports that help you demonstrate traceability between the work items stored in VersionOne 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 VersionOne. See Associating Requirements with Files for instructions on enabling this optional feature.

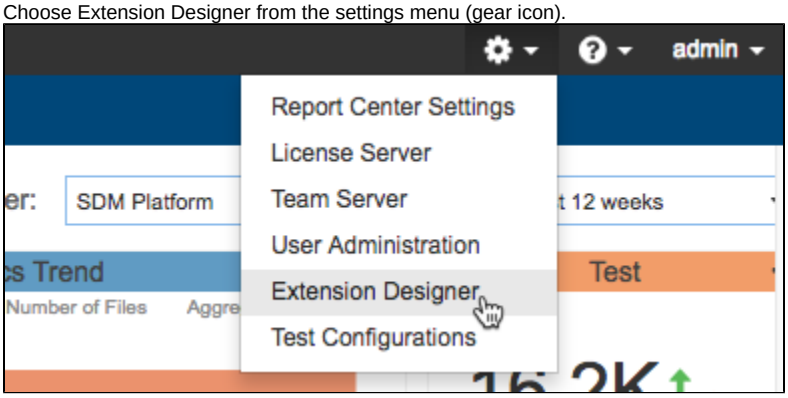
DTP interfaces that display and track traceability are enabled by deploying the External Application Traceability Report artifact shipped with the Traceability Pack. The Traceability Pack also includes the Sending Test Data to External Application flow, which automates part of the traceability workflow.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

Use DTP Extension Designer to deploy the External Application Traceability Report and the Sending Test Data to External Application flow to your environment. Verify that DTP is connected to VersionOne as described in the Connecting DTP to VersionOne 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).

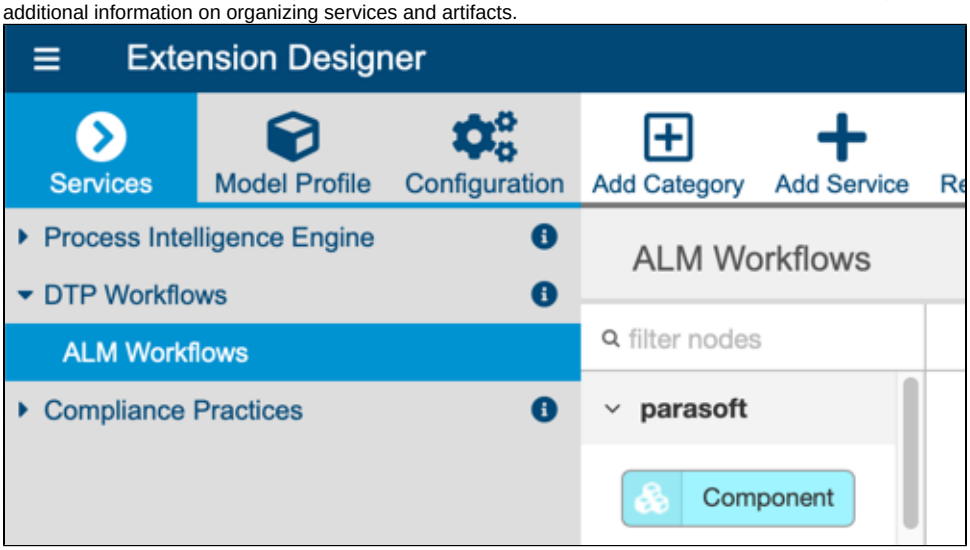


2. Click the **Configuration** tab to open Artifact Manager.
3. Click **Upload Artifact** and browse for the external-app-traceability-report-<version>.zip archive (also see Downloading and Installing Artifacts).
4. Click **Install** and a collection of assets and configuration files for enabling traceability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

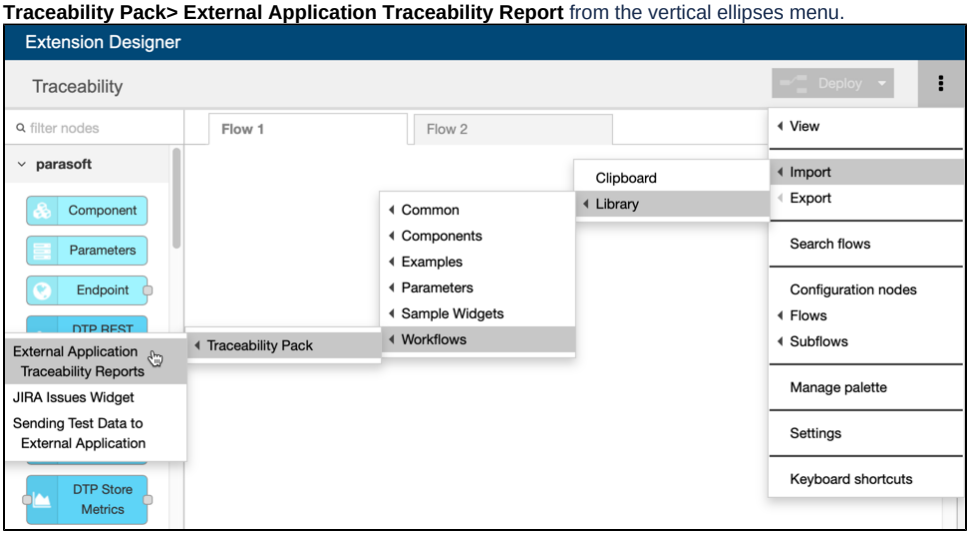
Deploying the External Application Traceability Report

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

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



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



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

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

Deploying the Sending Test Data to External Application Flow

This artifact sends test data to VersionOne 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 VersionOne integration. The out-of-the-box configuration uses default or commonly-used fields and work item types. If you customized your VersionOne system, however, then you can configure the following settings to align data in DTP with your custom configuration.

versionOneIssueUrl	Specifies the URL template for linking work items created in the DTP Violation Explorer and Test Explorer to work items in VersionOne. Default: `versionOneIssueUrl=<VERSIONONE_URL>/assetdetail.v1?number=<ID>`
versionOne.workItemType.defect	Specifies the type of work item to create in VersionOne when creating new defects from the DTP Violation Explorer and Test Explorer. This enables you to associate custom defect trackers you may have configured in VersionOne with work items created from DTP. By default, the property is not set. As a result, defect work items created in DTP are associated with task work items in VersionOne.
versionOne.workItemType.issue	Specifies the type of work item to create in VersionOne 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 VersionOne with work items created from DTP. By default, the property is not set. As a result, issue work items created in DTP are associated with task work items in VersionOne.

Usage

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

Manually Creating Defects and Issues in VersionOne

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 VersionOne assets in explorer views:

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

If the VersionOne story is closed, DTP will still create the artifact, regardless of whether the API or UI is used.

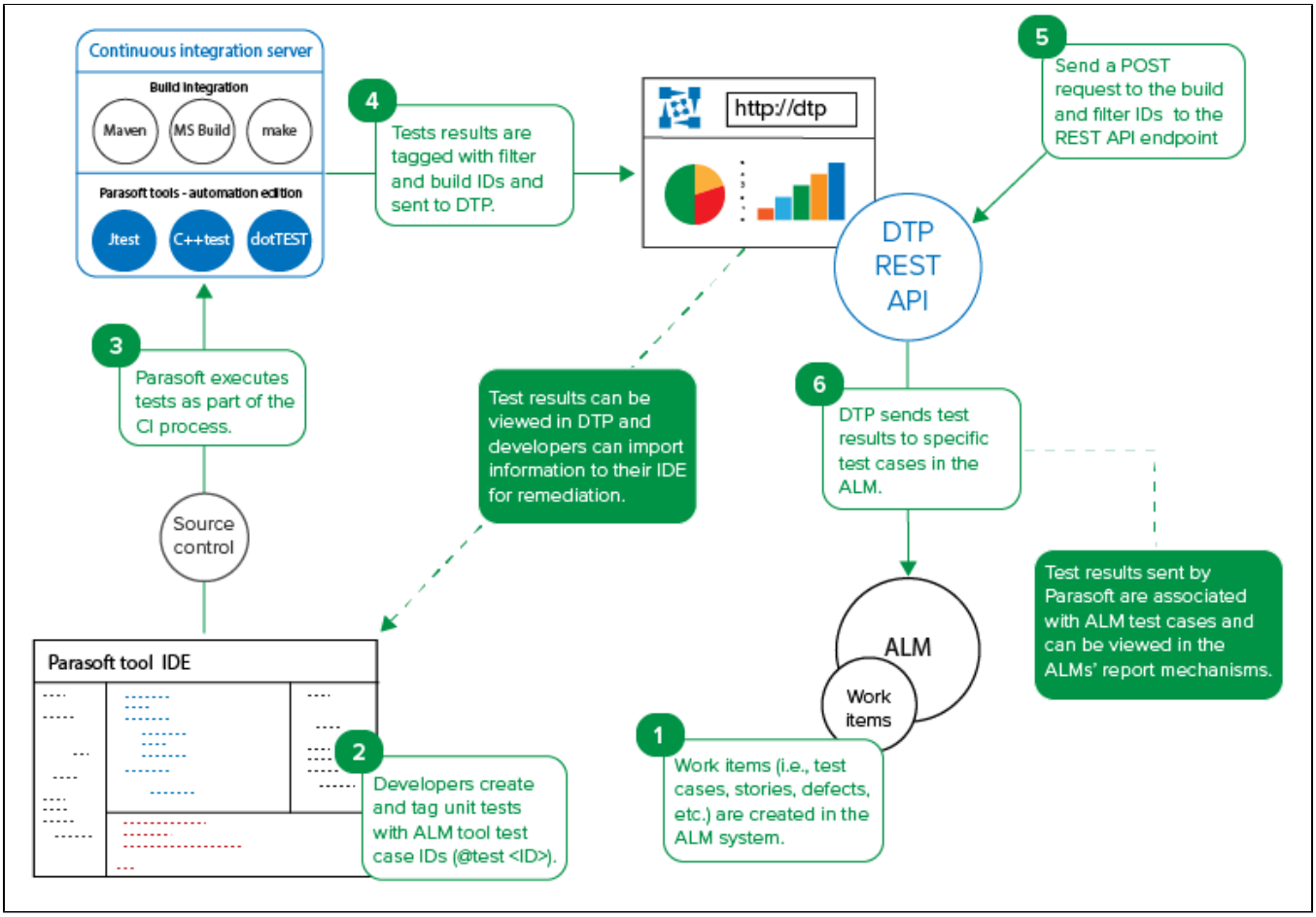
> **VersionOne terminology**
>
> "Issues" in VersionOne refer to work items that highlight items that require project-wide visibility. Issues can be represent anything that is impeding or might impede the project team's delivery. "Defects" are work items that identify a discrepancy between the expected and actual behavior of a completed feature. Defects can be scheduled into sprints/iterations the same way stories/backlog items are or, they can be tracked as a separate work queue outside of any sprint.

The following chart describes how assets are created in VersionOne:

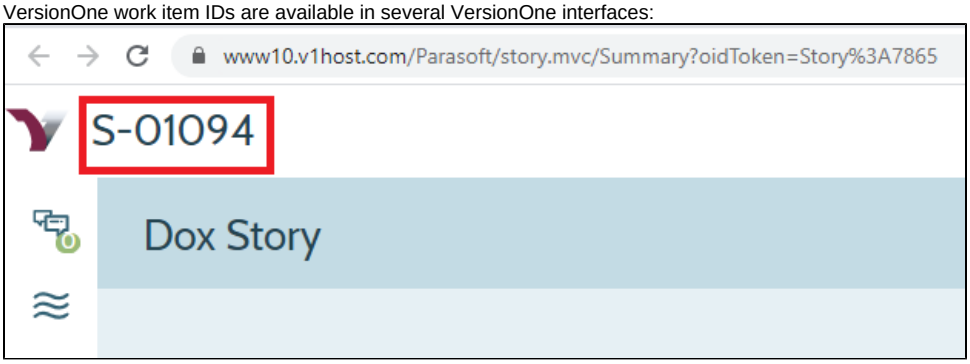
	Associated with @req annotation	**Associated with @test annotation**
Defect	• DTP creates assets in the Breaks Work Item field. • Links to the Test Explorer view	• DTP creates asset in the Breaks Work Item field. • Links to the Test Explorer view • Links to test in VersionOne
Issue	• DTP creates asset in the Blocking Issues field • Links to the Test Explorer	• DTP creates asset in the Blocking Issues field • Links to the Test Explorer view • Links to test in VersionOne

Sending Test Data to VersionOne

VersionOne work items are assets that represent a story, defect, or test set. Add the `@test` annotation and VersionOne test ID to the test code executed by your Parasoft tool to associate Parasoft tests with VersionOne tests. Add the `@req` annotation and VersionOne story ID in the test code to associate Parasoft tests with VersionOne stories. Refer to your Parasoft tool documentation for details on adding associations and the VersionOne documentation for information about getting work item IDs. The following diagram shows how you could implement an automated infrastructure for integrating Parasoft into your VersionOne environment:



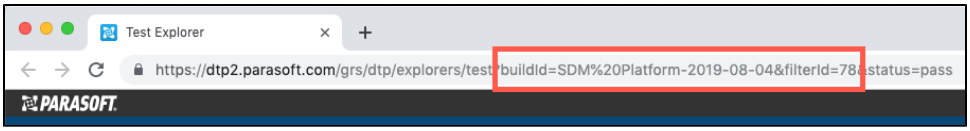
1. Create work items in VersionOne that will be associated with tests executed by Parasoft C/C++test, dotTEST, or Jtest.
2. In your test file, add the VersionOne work item IDs using the `@test` or `@req` annotation. See the C/C++test, dotTEST, or Jtest documentation for details on adding annotations.
    - Use the `@test <VersionOne test ID>` annotation in your tests to associate them with VersionOne tests.
    - Use the `@req <VersionOne story ID>` annotation in your tests to associate them with VersionOne stories.
    - VersionOne work item IDs are available in several VersionOne interfaces:



3. Execute your tests as part of the CI process. You can also manually execute the tests from your tool's desktop.
4. As part of the test execution, the tool 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 VersionOne when Data Collector receives the data from the Parasoft tool.
   You can also manually send a POST request to the DTP REST API endpoint to send results from the DTP database to VersionOne. Pass the DTP filter and build IDs as URL parameters in the API call:

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

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



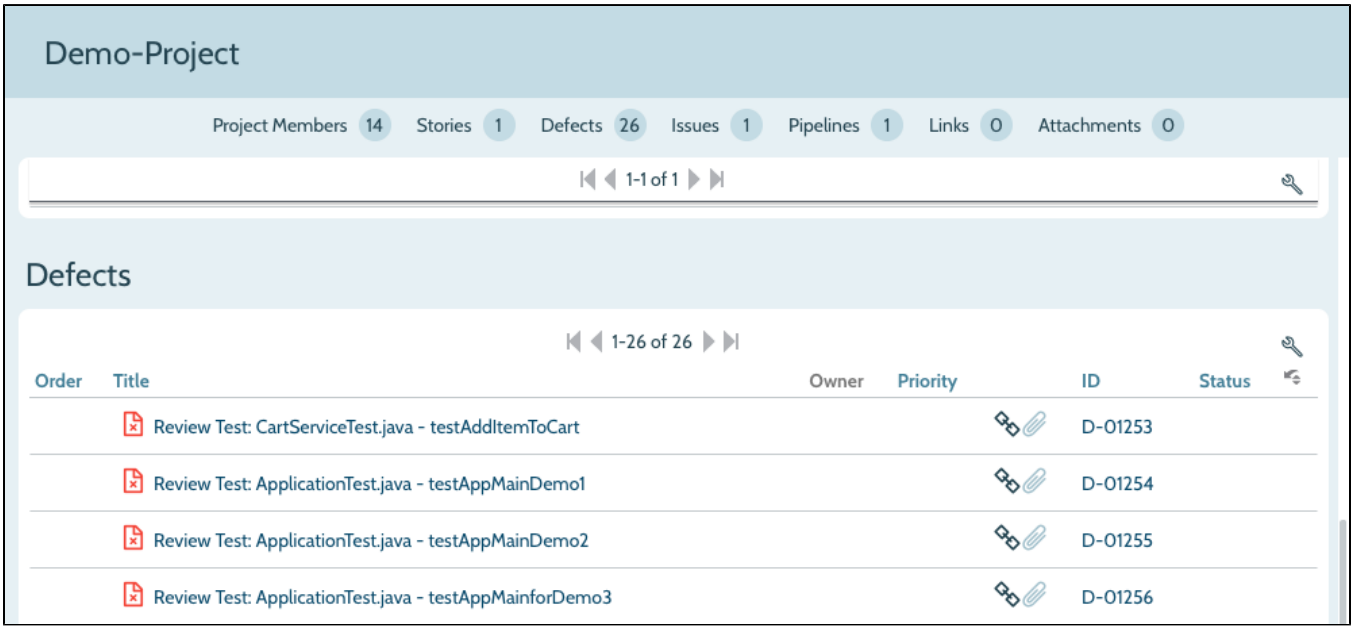
6. DTP will locate the test results that match the `filterId` and `buildId` parameters and send the data to the VersionOne work items. You should expect the following response:
    - When DTP locates results with an `@test <ID>`, it will search for unit test cases with a matching ID in VersionOne and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

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

```
{
        "createdTestSession":"DTPP-521"
    "created" : [
                "DTPP-519, testName = testBagSumAdd"
    ],
    "updated" : [
        "Test:1545 for AT-01053, testName = test_quoteGhsLine_Exp_Act_3",
        "Test:1546 for AT-01054, testName = test_quoteGhsLine",
        "Test:1554 for AT-01056, testName = test_quoteGhsLine_Exp_Act_10",
        "Test:7177 for S-01045, testName = test_quoteGhsLine_moreThanOne"
    ],
    "ignored" : [
                "MAGD-567, testName = testBagNegate",
        "QAP-512, testName = testTryThis3",
        "QAP-512, testName = testTryThis4",
        "MAGD-567, testName = testBagMultiply"
    ]
}
```

Viewing Results in VersionOne

After successfully sending the test data to from DTP, you will be able to view results VersionOne assets



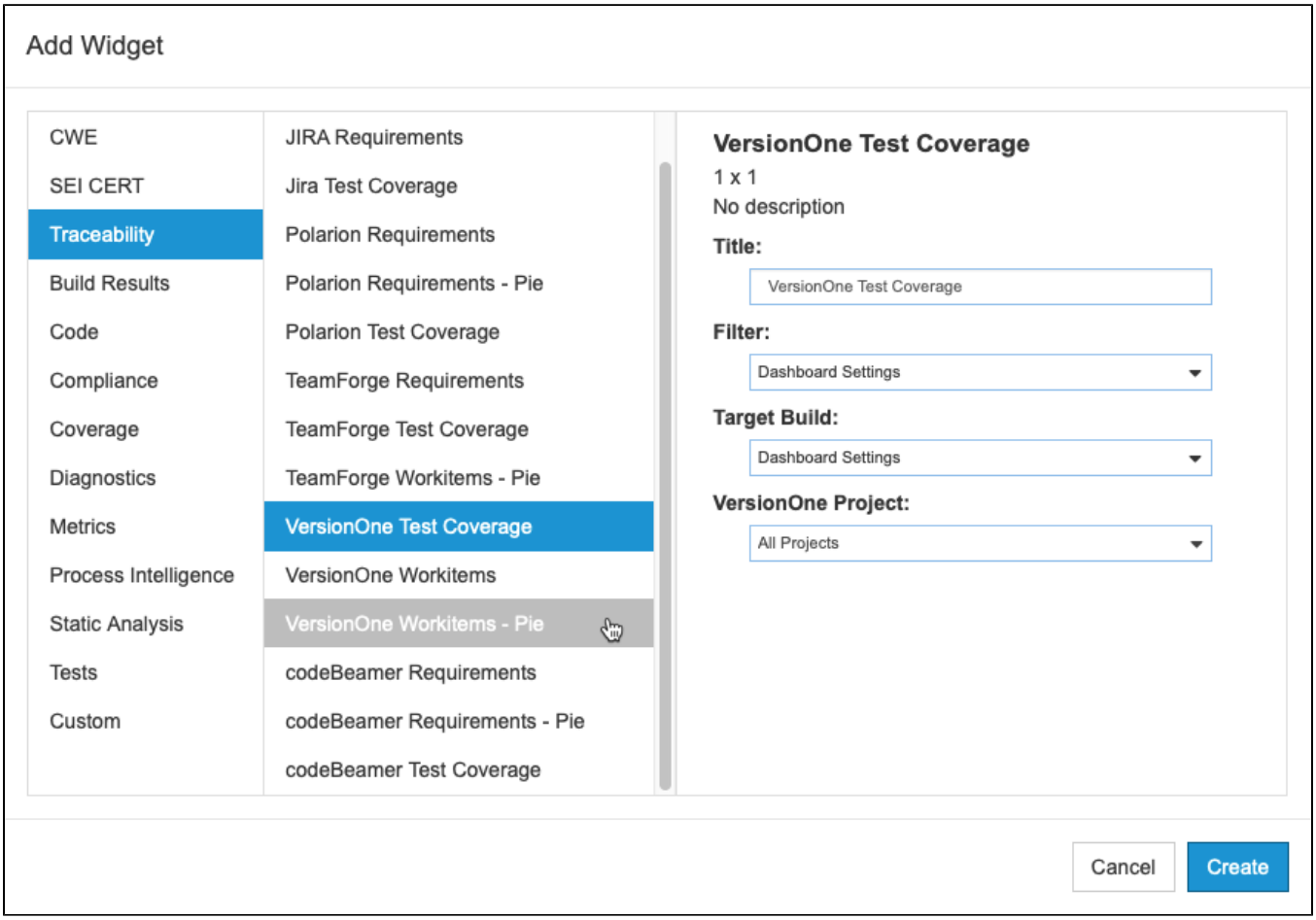
You can drill down into results to see details about the work item in DTP, which includes build and authorship information, as well as links back to the test or violation in DTP.

Viewing the Traceability Report

If the External Application Traceability Report has been deployed to your system (see Enabling the Traceability Report), you can add widgets to your dashboard to monitor traceability from work items 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.



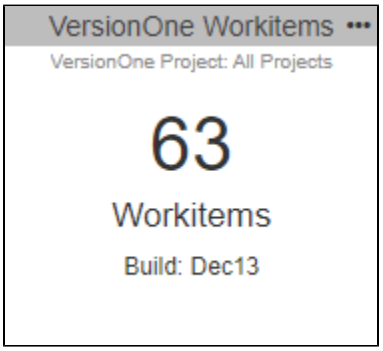
You can configure the following settings:

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 VersionOne project from the drop-down menu.

Requirements Widget

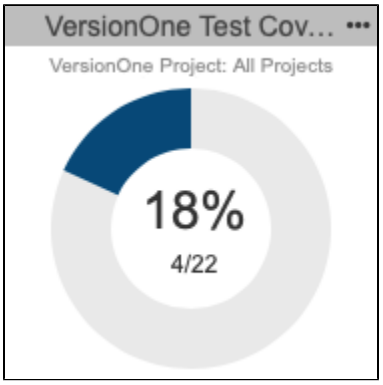
This widget shows the number of work items from the specified VersionOne project.



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.

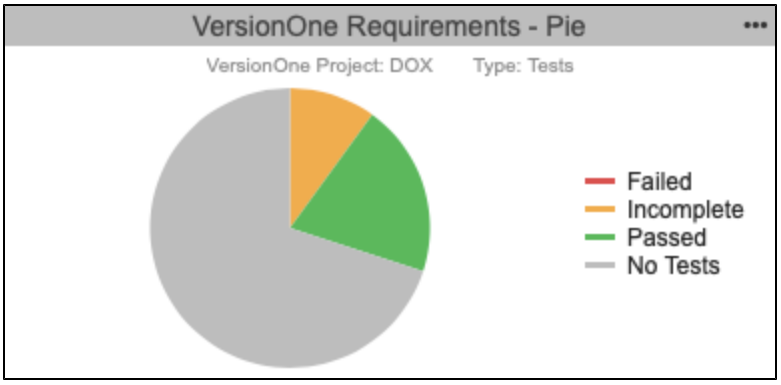
The colored-in segment represents the requirements covered by tests. Click on the segment to open the Requirement Traceability report filtered to the **With Tests** category.

Pie Widget

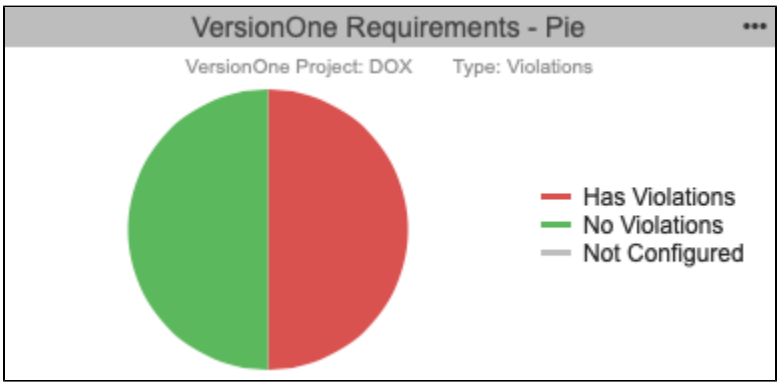
Unit testing, functional testing, static analysis, and peer reviews are common activities for verifying that work items have been properly and thoroughly implemented. This widget shows the overall status of the project work items in the context of those software quality activities. You can add a widget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

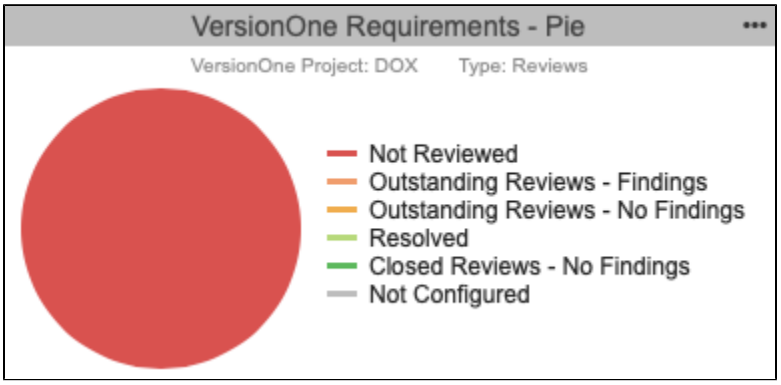
Requirements Implementation Status by Tests



Requirements Implementation Status by Violations

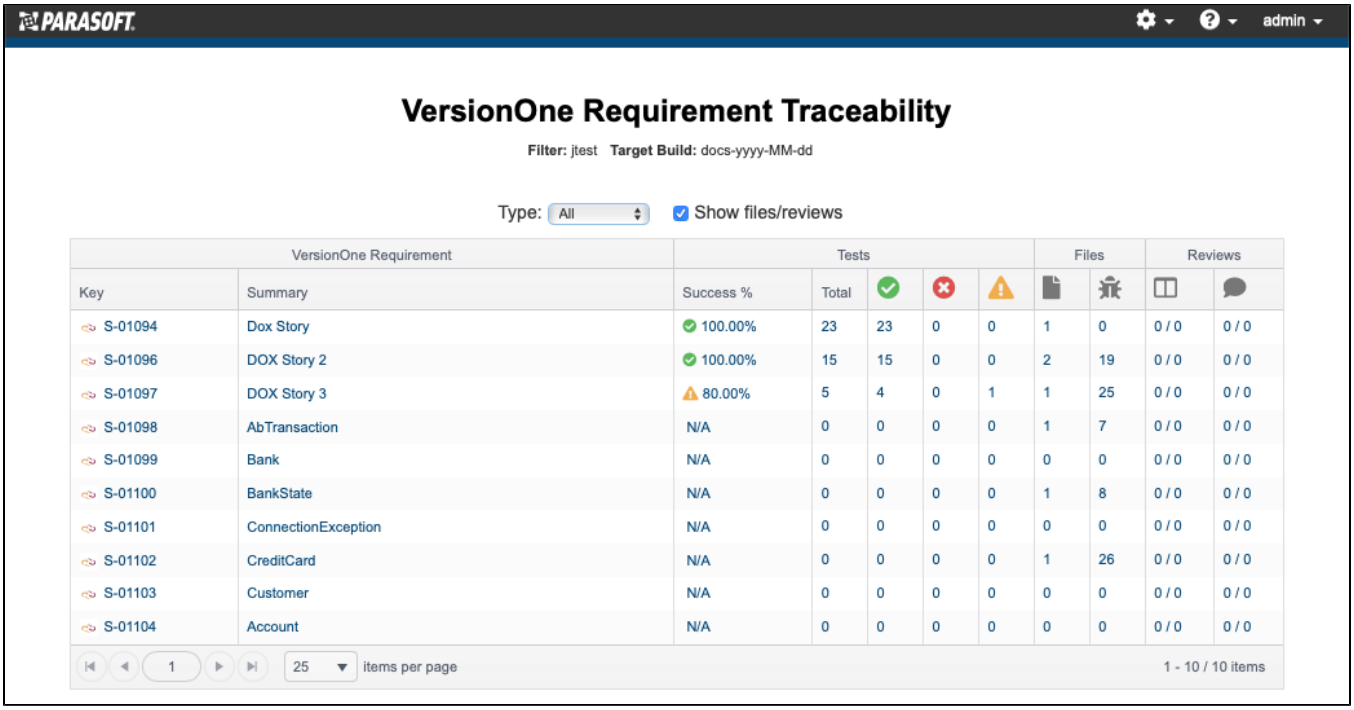


Requirements Implementation by Reviews



Understanding the Requirement Traceability

The report lists the VersionOne project work items and data associated with them.

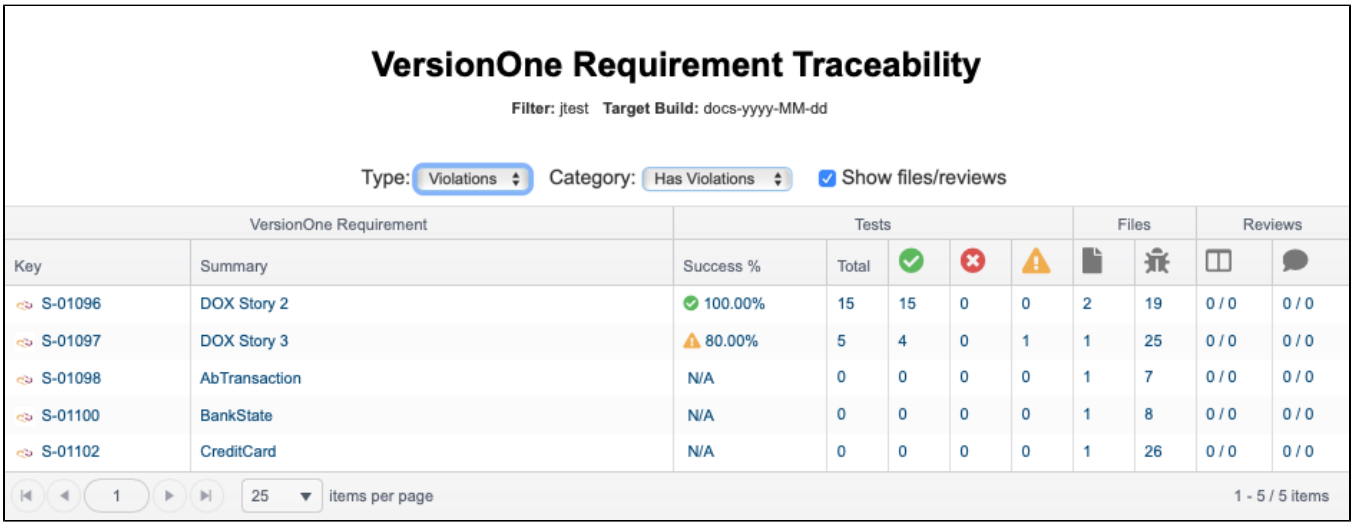


You can perform the following actions:

- 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 work item in VersionOne.
- Click on a link in the Summary column or one of the Test columns to view the test-related information associated with the work item 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 work item 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 work item in the Requirement Details Report.

Requirement Traceability Report by Type

Clicking on a section of the Pie widget opens a version of the report that includes only the quality activity type selected in the widget. 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 (see Enabling the Requirements Traceability Report). Disabling the Files and Reviews columns on this screen hides the related tabs in the Requirement Details report.

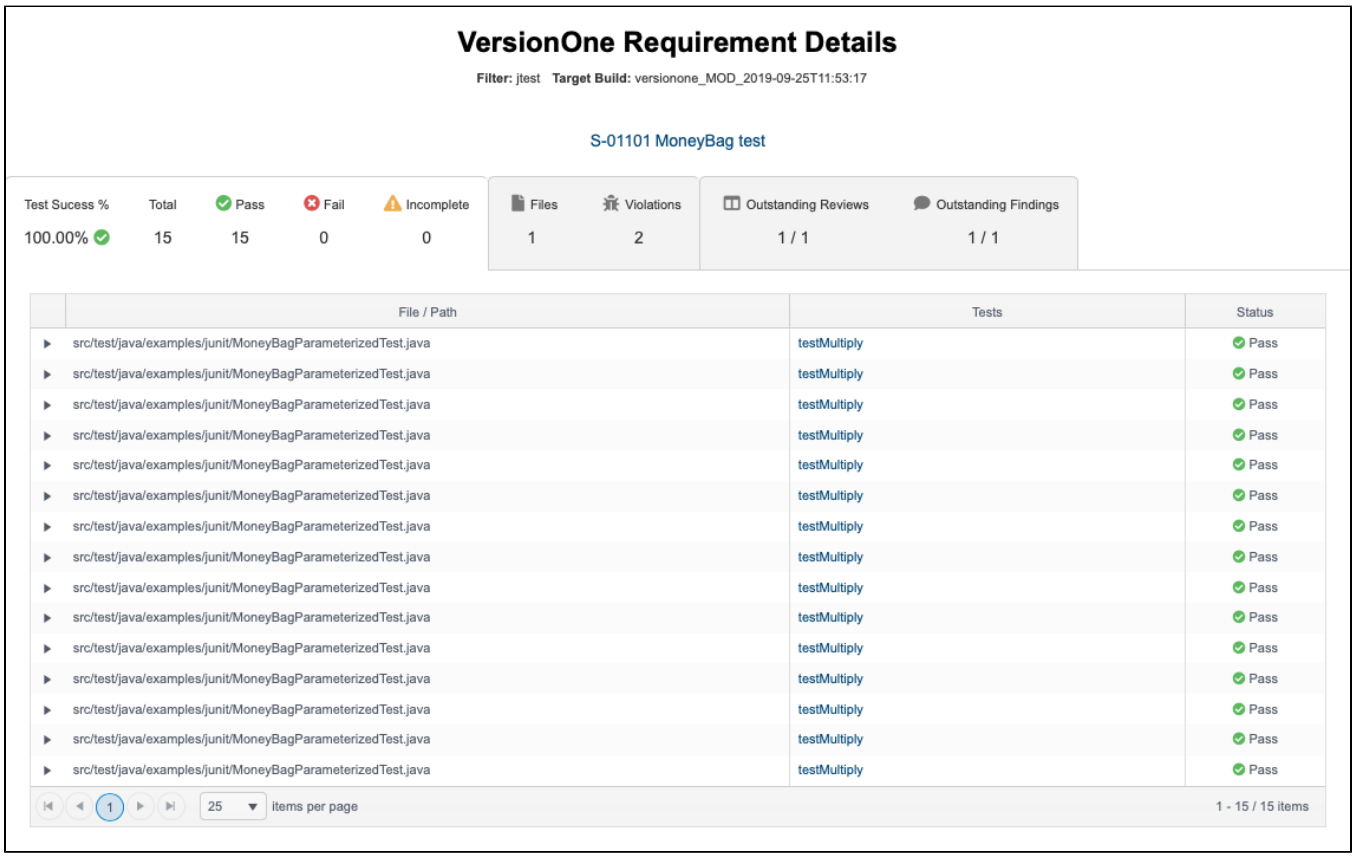


Understanding the Requirement Details Report

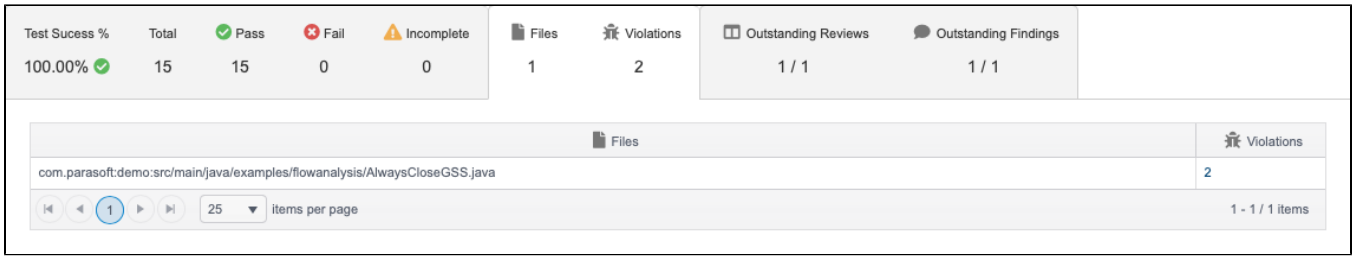
The Requirement Details report provides additional information about the files, static analysis findings, and tests associated with a specific VersionOne work item. Stories marked as `Dead` will not appear in the Traceability Report. The Traceability Report shows information about the following VersionOne work items:

- Requirements
- Defects

You can open this report by clicking on a work item in the main Requirement Traceability report.

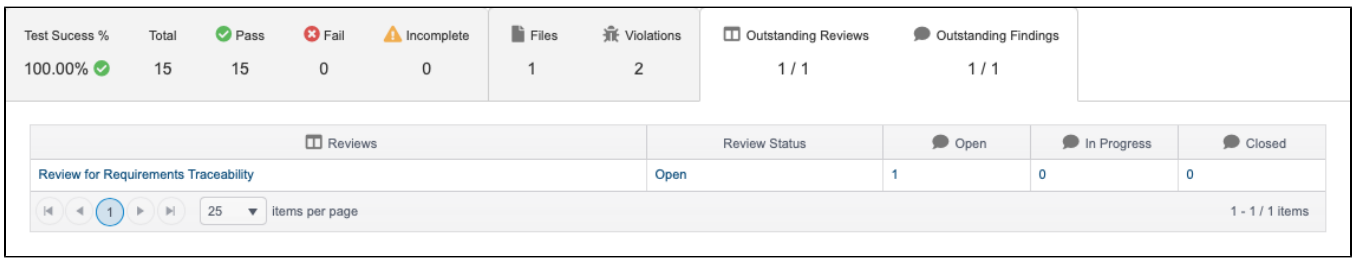


The first tab shows the results of the tests that were executed to verify the specific work item. Click on a test name to view the test in the Test Explorer.



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

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.

his 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.