

Working in regulated environments and ensuring compliance using Xray

- Overview
- Ways to ensure regulatory and compliance needs
 - Process
 - Implement your own-defined project organization, with fine control over project entities
 - Implement fine Access control and Permissions
 - Use custom defects with their own semantics and decide where to store them
 - Define what is a requirement and track them at multiple levels
 - Track the relevant requirements across multiple versions and/or environments
 - Implement Workflows on testing entities to have explicit control over the process
 - Sign using e-signatures to track explicit reviewal/approval and ensure data integrity
 - Facilitate reviewing, also during test execution
 - Use detailed, trackable and yet flexible Test Planning
 - Track testing tasks with full Assignment and Accountability
 - Enable Customization to fine-tune the tool to your overall rigorous process
 - Data
 - Record all meaningful testing evidence
 - Ensure data persistence, history and change tracking
 - Ensure no data tampering
 - Export data in human-readable formats and automate data snapshots
 - Risks
 - Handle risks with proper Risk Management and Risk-Based Testing
 - Mitigate risks by fostering clarity and facilitating collaboration, including on all testing activities
 - Trace relationship between entities, up to the code
 - Facilitated diagnosis and Root-Cause Analysis
 - Learning
 - Facilitated training resources to improve knowledge and ensure a seamless adoption
- Standards and regulatory requirements

Overview

Organizations working in highly regulated markets (e.g., medical/health, pharmaceutical, security, automotive, aerospace, defense), usually have to comply with specific standards or some regulatory requirements that apply to them and the products they are responsible for.

As an example, ensuring safety is essential for organizations developing medical devices.

A strict control over the development process (testing included), release process, and its maintenance is required. Risk management is core to all activities.

In order to have a proper auditable process in place, that is compliant with high-demanding regulatory requirements, we need to:

- Ensure persistence of historical data & change tracking
- Ensure that historical data cannot be tampered
- Ensure that certain items can only be modified if people are allowed to do so
- Ensure it's possible to identify who is responsible for what
- Identify the relationship between entities
- Ensure testing has been performed and that acceptance criteria has been covered
- Proof of the exact testing/checks that were performed
- Ensure diagnosis is facilitated and can be easily performed
- Ensure effective risk management is in place

Ways to ensure regulatory and compliance needs

Process

Overall, the process must be adjustable, controllable, documentable, clear, and tasks must be repeatable and auditable. Diagnosis should also be facilitated, to be able to understand events. On parallel, we must be able to properly manage risks.

Among others, we need to:

- implement a tailored, auditable, and controlled process (e.g., by configuring the project organization, by defining the semantics of entities, by enforcing rules & workflows for testing entities)
- support planning, including at testing level (i.e., using Test Plans)
- ensure tasks, including the ones derived from testing, are complete (i.e., by using tailored Xray issues)
- verify and analyze software requirements from multiple perspectives
- support unit, integration and system testing (i.e., by supporting diverse test automation frameworks alongside manual and exploratory testing)
- enable regression testing (i.e., by having tailored Test Sets or Test Plans for that)
- ensure software verification is complete (i.e., using coverage information)
- document released versions, including testing effort (i.e., by generating documents containing all the relevant data, including testing evidence)

- [facilitate software problem resolution processes \(i.e., by using reports such as the Traceability one\)](#)

Implement your own-defined project organization, with fine control over project entities

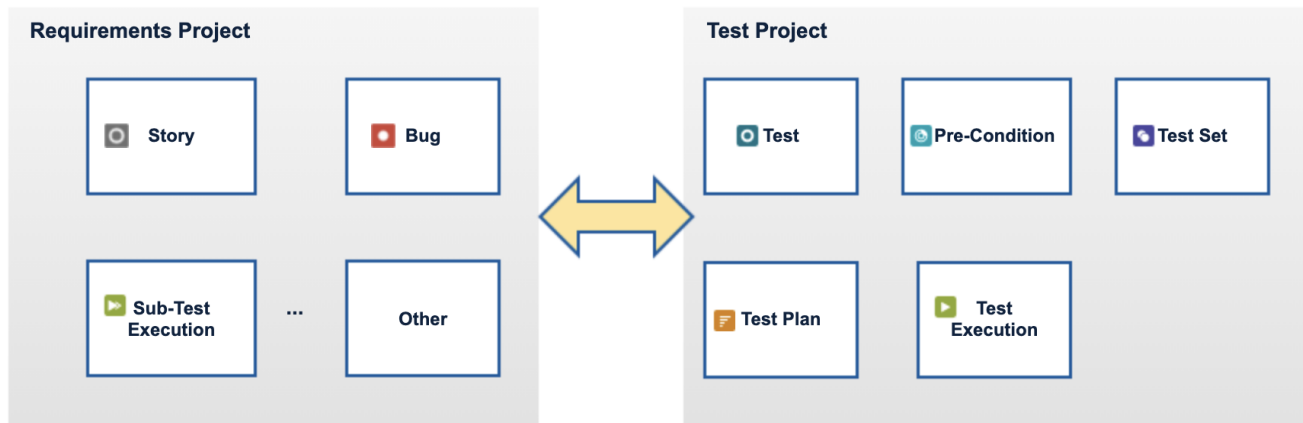
Xray supports different ways for organizing your project related items. This enables teams to adapt usage of the tool with the way they need or to the way they are organized functionally. But more than that: it enables teams to enforce or apply different rules to different entities.

The common scenario is to have an all-in-one approach where standard issue types (e.g. Story, Bug, Task) live alongside testing issue types (e.g. Test, Test Execution, Test Plan).

However, some teams prefer to have testing entities on a separate Jira project to have finer control over the process. Other teams prefer to have defects on a distinct project, or even on a remote Jira instance for security/business reasons.

All these possible project organization scenarios are possible and allow teams to apply more restricted rules on the related entities.

In the following scenario, two Jira projects are used: one mainly for the standard development/coding tasks and another one for testing purpose. This one of multiple configuration possibilities you can implement.



Even though you can be compliant with an all-in-one approach, Xray provides you the flexibility to even better suit your needs.

Learn more:

- [Project Organization Use Cases](#)

Implement fine Access control and Permissions

Organizations working in regulated environments need to be able to control the access and the permissions on project artefacts, including the ones derived from testing activities.

Access and permissions for Jira issue based entities (i.e. Test, Pre-Condition, Test Set, Test Plan, Test Execution, Sub-Test Execution) are handled in the same way as for any other issue type in Jira.

It's also possible to restrict execution of Test Runs to its assignee, so that only the assignee can update its contents.

Learn more:

- [Advantages of using Jira issues](#)
- [Configuration to restrict execution of tests to the Test Run's assignee](#)

Use custom defects with their own semantics and decide where to store them

Semantically, and to make it clear and be able to apply different rules, it can make sense to have different issue types than the traditional "Bug."

In Xray, it's possible to define the issue types to be handled as "defects." Therefore, whenever reporting a bug, the team can use the Bug issue type or any other custom issue type where they need.

Defects can also be covered explicitly with tests (if configured as "requirements"/coverable issue types), for explicit verification of the underlying bug fixes.

Issue Type Mapping

Requirements and Defects are two concepts commonly related to Tests. A Requirement represents a singular documented fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended and a Defect.

The screenshot displays the Jira Issue Type Mapping interface, which is organized into three main columns:

- Available Issue Types:** A list of issue types that can be mapped, including Test, Test Set, Test Execution, Pre-Condition, Test Plan, Incident, Change, Sub-task, and Technical task.
- Requirement Issue Types:** A list of issue types that can be mapped to requirements, including Epic, New Feature, Story, Improvement, and Incident. The 'Incident' type is highlighted with a red border.
- Defect Issue Types:** A list of issue types that can be mapped to defects, including Bug, Incident, and Problem. This entire column is highlighted with a red border.

Defects can be reported on the same project alongside other project issues, or can be reported on a separate Jira project. Defects can also be reported on a separate but connected Jira instance, if you wish to do so.

Learn more:

- [Issue Type Mapping](#)
- [Requirements and Defects](#)

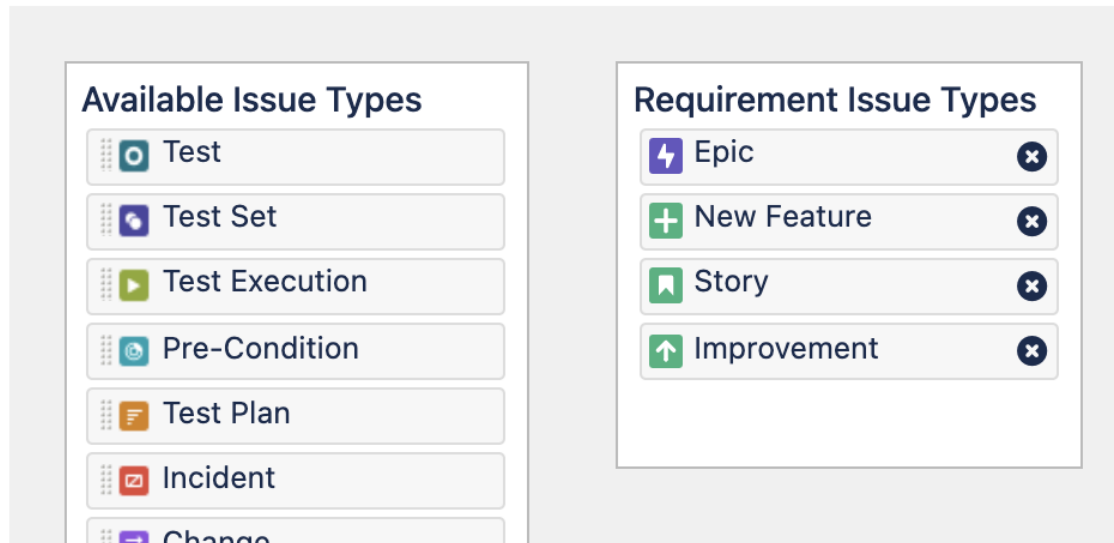
Define what is a requirement and track them at multiple levels

Projects usually have multiple levels of requirements (e.g. Epic>Story). Some broad/complex needs may be addressed in smaller requirements. Value can therefore be decomposed in these different layers and it's important to track it and these layers.

In Xray, a "requirement" is any issue that can be covered with tests, no matter their type. The team can define which issue types can be handled as such, even their own custom issue types (e.g., "Feature").

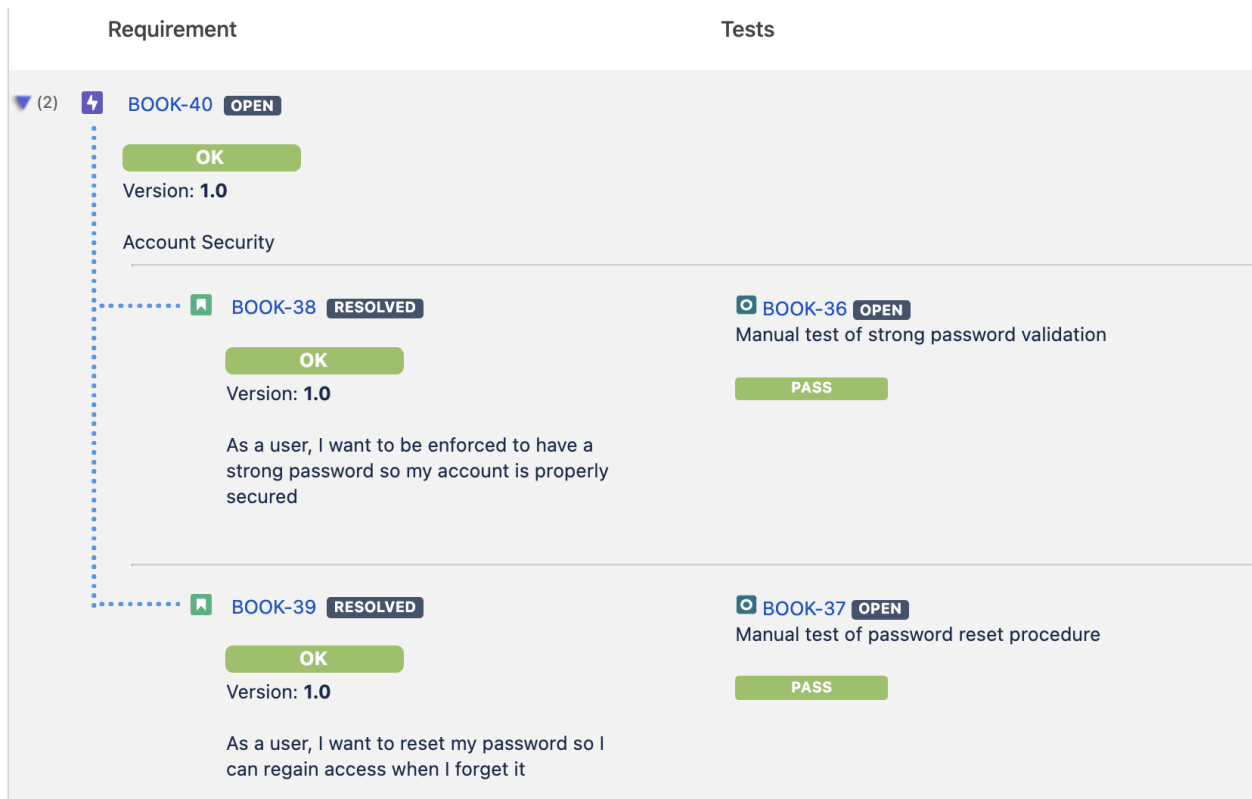
Issue Type Mapping

Requirements and Defects are two concepts commonly related to Tests. A Requirement in a computer program or system that produces an incorrect or unexpected result is called a Defect.



Xray provides multiple levels of requirements. It also provides different ways to define this hierarchical relationship (e.g., using issue links, or sub-tasks)

Epic and Story issues, usual in Agile environments, can be covered directly by testing. Coverage made on the Story issues can automatically be tracked on the related Epic.



Learn more:

- [Issue Type Mapping](#)
- [Requirements and Defects](#)

Track the relevant requirements across multiple versions and/or environments

Requirements exist beyond the scope of a specific version. Besides, a specific requirement may be used in different contexts (e.g., browsers, mobile devices).

Therefore we need to be able to track the status of requirements throughout time/versions and also on the different scenarios where they will be used. This (coverage) status is based on the integrated testing results, include the ones from test automation.

We may want to perform this analysis at high-level (i.e., project), or for a/some specific requirement(s).

Xray allows you to answer questions such as:

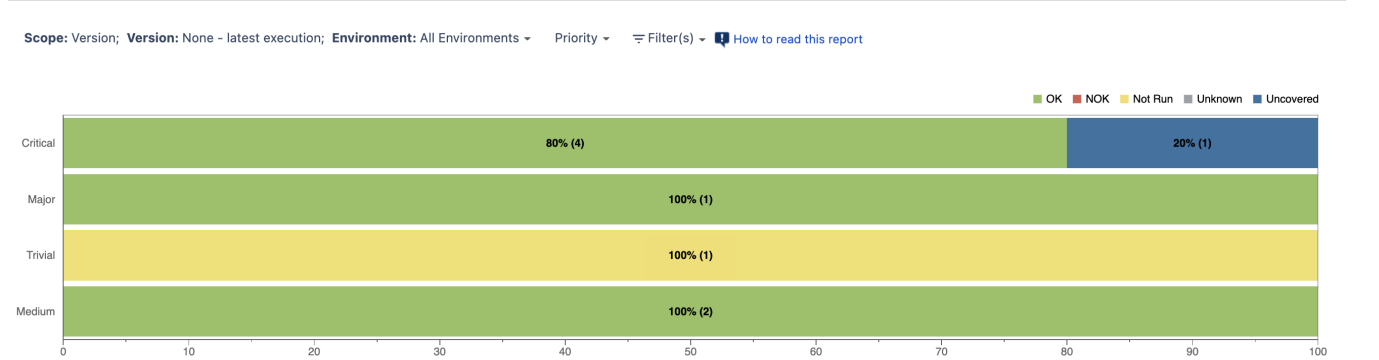
- Are the requirements covered by any tests whatsoever?
- What were the requirements tested for these version? And what's their status based on the related testing? What's their completeness?
- How are the requirements implemented in an earlier version, on this new/other version?

The coverage status for requirements in Xray is multidimensional; in other words, we can analyze requirements from multiple perspectives: the latest testing results for version, or on a given environment, or just considering the results performed in the scope of a Test Plan.

Let's see some examples.

Accordingly with the latest reported results, all medium and major requirements are OK. However, we can see that one requirement is not covered. We can then drill-down if we wish, by clicking on the chart bar.

Overall Requirement Coverage Report [Switch report](#)

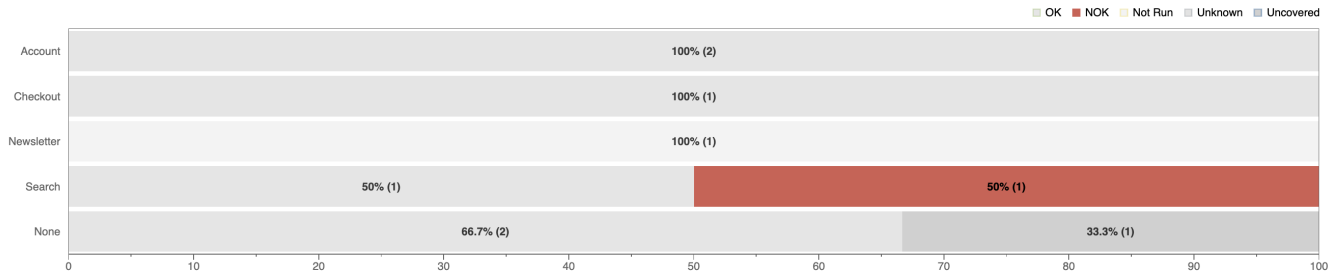


On version 1.0, 50% of the requirements related to the "Search" component are having problems. We can see that the epic BOOK-8 is 66,7% complete, because one of the related stories (BOOK-9) has 1 test currently failing on version 1.0.

Some related stories (BOOK-10, BOOK-11) are OK though.

Overall Requirement Coverage Report [Switch report](#)

Scope: Version; Version: 1.0; Environment: All Environments Component/s Filter(s) How to read this report



Requirements with Component/s "Search" and Status "NOK"

[View Issues](#)

Show 10 entries

| | | Search | | | | |
|---------|--|-------------|--------------|--------------|---------------|--------------|
| Key | Summary | Total Tests | Tests Passed | Tests Failed | Tests Unknown | Completeness |
| BOOK-8 | As a visitor, I can manage my Shopping Basket | 3 | 2 | 1 | 0 | 66.7% |
| BOOK-11 | As a visitor, I can view all the books in my shopping basket | 1 | 1 | 0 | 0 | 100% |
| BOOK-10 | As a visitor, I can remove books from my shopping basket | 1 | 1 | 0 | 0 | 100% |
| BOOK-9 | As a visitor, I can add books to my shopping basket | 1 | 0 | 1 | 0 | 0% |

Learn more:

- Coverage Analysis
- Overall Requirement Coverage Report

Implement Workflows on testing entities to have explicit control over the process

Teams can implement workflows on all Jira issue based entities (i.e. Test, Pre-Condition, Test Set, Test Plan, Test Execution, Sub-Test Execution).

This will allow them to implement rigorous processes, including on testing entities, leveraged on the flexibility of Jira workflows.

Additionally, Xray provides the ability to further restrict usage of some these entities.

Possible use cases:

- have workflow status to review test specifications
- implement an approval mechanism, having one or more approvers
- make items "read-only" when transitioned to a certain workflow status by setting a Jira property ("jira.issue.editable")
- restrict usage of Tests in a certain workflow status (see how)
- disallow executions for Test Executions in a specific status (see how)
- reopen/review Tests upon changes on related requirements (example)
- enforce that requirements cannot transition to a given status, unless they are covered or even that those tests are successful (example)

Learn more in:

- Using Jira workflows for testing purpose

Sign using e-signatures to track explicit review/approval and ensure data integrity

Electronic/digital signatures are one of the common mechanisms employed to ensure that document, for example, has been reviewed by one or more people and that no changes happened on the signed document meanwhile.

Xray issue based entities (i.e. Test, Pre-Condition, Test Set, Test Plan, Test Execution, Sub-Test Execution) can all be digitally signed, using one of many available Jira apps for that purpose. This is something unique, as it provides full control over the core testing activities; not only Tests can be digitally signed, also Test Sets, or even Test Plans.

Learn more:

- [available Jira apps for e-signatures](#)

Facilitate reviewing, also during test execution

Review during or after test execution can be implemented.

In Xray, similarly to other issues, Test Execution issues have an assignee. We can use that as the main responsible for the Test Execution, perhaps for defining the list of tests to be run and then for reviewing its results.

The Tests to be run on Test Executions (i.e., the Test Runs) can have their own specific assignee. Therefore, we can have a Test Execution responsible and the individual testers performing the identified testing.

This Test Execution as a whole is assigned to Petr Gonzalez but one of the Test Runs is assigned to another user (Bruno Conde). This can be used as means to have one person responsible for the Test Execution and reviewing the related reported results, while at the same time allowing for someone else to perform the actual testing.

Bookstore / BOOK-31

Test Execution for Test Plan BOOK-30

Edit

Comment

Assign

More

Start Progress

Resolve Issue

Close Issue

Admin

Details

Type: Test Execution

Priority: Major

Affects Version/s: None

Component/s: None

Labels: default

Test Plan: BOOK-30

Test Environments: None

Revision: 65d64uyg7t6r

Description

Click to add description

Tests

Add Tests

Overall Execution Status

14 PASS 1 FAIL 1 TODO

Total Tests: 16

Filter(s)

Apply Rank

Test Type #Req #Def Assignee Status

Rank

Key

Summary

Test Type

#Req

#Def

Assignee

Status

16

BOOK-36

Manual test of strong password validation

Manual

2

0

Bruno Conde

TODO

15

BOOK-37

Manual test of password reset procedure

Manual

2

0

Petr Gonzalez

PASS

Xporter

Template: Basic Release Notes

Output format: DOCX

Export

People

Assignee: Petr Gonzalez

Assign to me

Reporter: Xpand IT Admin

Votes: 0

Watchers: 1 Stop watching this issue

Dates

Created: 25/Feb/19 2:06 PM

Updated: Just now

Begin Date: 18/Feb/19 2:03 PM

End Date: 22/Feb/19 2:03 PM

Agile

Active Sprint: Sprint 1 ends 18/Mar/19

View on Board

This enables the implementation of a reviewal process, where Test Execution related results can be reviewed after they're reported; the Test Execution can be transitioned to a status (e.g., "PENDING_REVIEW") and then transitioned to another one (i.e., "CLOSED") when it's reviewed.

We can also create some specific Test Run custom field (e.g., "Reviewer Comment") and make it required for all runs, or just for the ones related to manual scripted test cases.

[illegible]

During execution of a test and its revision, we may find it useful to report a step as having an acceptable failure (e.g., "IRRELEVANT_FAIL"). We can then decide to mark the overall Test Run with a status (e.g., "FAIL_DISCARD") that won't mark the related requirement as being NOK, such as UNKNOWN or even OK if we want.

Test Statuses

Manage custom Test Run statuses for Xray

| Name | Description | Final | Color | Requirement Status |
|--------------|--|-------------------------------------|-------|--------------------|
| PASS | The test run has passed | <input checked="" type="checkbox"/> | | OK |
| TODO | The test run has not started | <input type="checkbox"/> | | NOTRUN |
| EXECUTING | The test run is currently being executed | <input type="checkbox"/> | | NOTRUN |
| FAIL | The test run has failed | <input checked="" type="checkbox"/> | | NOK |
| ABORTED | The test run was aborted | <input checked="" type="checkbox"/> | | NOTRUN |
| BLOCKED | | <input type="checkbox"/> | | NOTRUN |
| PENDING | | <input checked="" type="checkbox"/> | | UNKNOWN |
| FAIL_DISCARD | There was a minor issue but nothing relevant | <input type="checkbox"/> | | UNKNOWN |

Test Step Statuses

Manage custom Test Step statuses for Xray

| Name | Description | Color | Test Status |
|-----------------|---|-------|--------------|
| PASS | The test step has passed | | PASS |
| TODO | The test step has not started | | TODO |
| EXECUTING | The test step is currently being executed | | EXECUTING |
| FAIL | The test step has failed | | FAIL |
| SKIP | The test step has been skipped. | | PASS |
| IRRELEVANT_FAIL | There was a minor issue with this step but nothing relevant | | FAIL_DISCARD |

[Bookstore](#) / [Test Plan: BOOK-30](#) / [Test Execution: BOOK-31](#) / [Test: BOOK-36](#)

Manual test of strong password validation



[Return to Test Execution](#)

[Previous](#)

Execution Status FAIL_DISCARD

Started On: 24/Jun/19 11:29 AM

Finished On: -

Assignee: Bruno Conde

Versions: 1.0

Executed By: Xpand IT Admin

Revision: 65d64uyg7t6r

Tests
environments: -

Comment

[Preview Comment](#)

Execution Defects (0)

[Create Defect](#)

[Create Sub-Task](#)

[Add Defects](#)

Execution Evidence (0)

[Add Evidence](#)

Execution Details

Test Description

Test Issue Links (1)

tests

[BOOK-36](#) As a user, I want to be enforced to have a strong password so my account is properly secured

[RESOLVED](#)

Custom Fields

Test Steps (4)

| | | | |
|---|--|---|---|
| 1 | Action Open the Change Password screen by selecting option "My Profile > Password" | Data None | Expected Result None |
| | Actual Result | Comment | Defects (0) Evidence (0) Step State PASS |
| 2 | Action Fill the password fields with data | Data Current Password: passw0rd New Password: p4ssw0rd Confirm New Password: p4ssw0rd | Expected Result Error: "Current password is incorrect" |
| | Actual Result | Comment | Defects (0) Evidence (0) Step State IRRELEVANT_FAIL |

Use detailed, trackable and yet flexible Test Planning

Even though plans can change (especially in an Agile context), being able to define a plan, no matter its level of detail, is something usually required.

Some testing activities, including verification, can be planned upfront.

Xray provides an entity specifically for this purpose: the Test Plan. The Test Plan can be used more strategically, to define and track and assign testing activities to be performed.

In Xray, teams can have one or more Test Plans to track different testing efforts, for example related with different types of tests, or different risks, or even explicitly for regression testing.

The Test Plan can be as detailed as needed and documents can be added and linked to it. A Test Plan is also assigned to a user; it can be prioritized, labelled, and be explicitly one work item of the Sprint or of the Kanban board.

In the following example, we can see the Test Plan being tracked as an item and handled as part of the Sprint, similarly to other on-going project issues.

Sprint 1

QUICK FILTERS: [Only My Issues](#) [Recently Updated](#)

To Do

CALC-838

As a user, I can calculate the sum of 2 numbers

v3.0 - NOTRUN

CALC-848

Sub Test Execution for CALC-838

CALC-954

Sub Test Execution for CALC-838

In Progress

CALC-970

plan for v3.0, sprint 1

CALC-962

As a user, I can calculate the multiplication of 2 numbers

v3.0 - OK

CALC-966

Sub Test Execution for CALC-962

CALC-967

As a user, I can subtract 2 numbers

v3.0 - NOK

CALC-969

Sub Test Execution for CALC-967

Done

CALC-964

As a user, I can calculate the division of 2 numbers

v3.0 - OK

CALC-964

Sub Test Execution for CALC-961

Learn more:

- [Planning Tests](#)
- [Tips for planning tests](#)
- [Some examples of using Test Plans in Scrum context](#)
- [Agile Board Enhancements](#)

Track testing tasks with full Assignment and Accountability

Xray issues (i.e. Test, Pre-Condition, Test Set, Test Plan, Test Execution, Sub-Test Execution) can be assigned and watchers can be added to them.

That means, for example, that whenever a Test is modified, you can get notified.

Bookstore / BOOK-24

Test visitors can remove books from their shopping basket

Edit

Comment

Assign

More

Start Progress

Resolve Issue

Workflow

Admin

Details

Type: Test

Priority: Trivial

Affects Version/s: None

Component/s: None

Labels: default

Test Repository Path: Shopping Basket

Description

Click to add description

Test Details

Type: Cucumber

Scenario Type: Scenario

Scenario: Given I am on shopping basket page
When I click the remove from basket button for a listed book
Then the book is deleted from my shopping basket

Xporter

Template: Basic Release Notes

Output format: DOCX

Export

People

Assignee: Bruno Conde

Assign to me

Reporter: Petr Gonzalez

Votes: 0 Vote for this issue

Watchers: Stop watching this issue

Dates

Created: 25/Feb/19 1:39 PM

Updated: Just now

Ensure issues get done by assigning them to users and tracking them using workflows (more info ahead), no matter if you're dealing with the specification of a test case or the execution of a bunch of tests within a given Test Execution.

Individual Test Runs can also be assigned and it's possible to notify the assignee using traditional email notifications, just like any other issue in Jira.

Learn more:

- [Email Notifications](#)

Enable Customization to fine-tune the tool to your overall rigorous process

For defining a more rigorous process, teams need the ability to customize usage of the tool.

This includes, among others, the ability to:

- add custom fields on any testing entity, including all Jira issue based entities (i.e. Test, Pre-Condition, Test Set, Test Plan, Test Execution, Sub-Test Execution) and also on Test steps and on Test Runs
- implement additional customizations (e.g., using ScriptRunner, Automation for Jira, Cprime Power Apps)
- integrate with other Jira apps to widen the feature set provided by Xray itself

Learn more:

- [Advantages of using Jira issues](#)
- [Integration with ScriptRunner](#)
- [Integration with Automation for Jira](#)
- [Integration with Cprime Power Scripts, Power Custom Fields, and Power Actions](#)

Data

Record all meaningful testing evidence

Teams can follow different approaches/styles for testing: scripted (i.e. test cases and "automated" test scripts) and exploratory.

No matter the testing approach, Xray can provide visibility of testing results, including evidence, in one place: Jira.

Xray allows you to identify:

- version and revision of the SUT
- environment (e.g., browser, device)
- test result/status
- detailed results (e.g. steps)
- screenshots, videos or any other file-based evidence
- execution logs
- any reported comments

Whenever manually executing scripted test cases, we can report results at the step level, and attach some screenshot as evidence, for example:

Atlas Mobile App / Test Plan: ATLAS-17 / Test Execution: ATLAS-19 / Test: ATLAS-8
Manual test of strong password validation

Dataset Return to Test Execution Previous Next

Test Details

Custom Fields

Test Description

Test Issue Links

Test Steps

| 1 | Action | Data | Expected Result |
|---|---|--|---|
| | Open the Change Password screen by selecting option "My Profile > Password" | None | None |
| | Actual Result | Comment | Defects (0) Evidence (0) Step State PASS |
| 2 | Action | Data | Expected Result |
| | Fill the password fields with data | Current Password: passw0rd New Password: p4ssw0rd Confirm New Password: p4ssw0rd | Error: "Current password is incorrect" |
| | Actual Result | Comment | Defects (0) Evidence (0) Step State PASS |
| 3 | Action | Data | Expected Result |
| | Close error message and fill again the password fields with data | Current Password: P4ssw0rd New Password: password Confirm New Password: password | Error: "New password is too simple" |
| | Actual Result | Comment | Defects (0) Evidence (0) Step State PASS |
| 4 | Action | Data | Expected Result |
| | Close error message and fill again the password fields with data | Current Password: P4ssw0rd New Password: P4ssw0rd Confirm New Password: P4ssw0rd | Information message: "Password successfully changed" |
| | Actual Result | Comment | Defects (1) Evidence (1) Step State FAIL |

Screenshot_1626039796...

Activity

Detailed information from multiple test automation frameworks can also be imported to Xray, and tracked similarly to other tests.

This includes automation frameworks such as Robot Framework, where you can see the detailed keyword-level information.

Robot / Test Plan: ROB-12 / Test Execution: ROB-17 / Test: ROB-18
Valid Login

Export Test as Text Return to Test Execution Next

Execution Status **PASS**

Started On: 14/May/20 4:42 PM Finished On: 14/May/20 4:42 PM

Assignee: Administrator Versions: -
Executed By: Administrator Revision: -
Tests environments: -

Comment Preview Comment Execution Defects (0) Create Defect Create Sub-Task Add Defects Execution Evidence (0) Add Evidence

Execution Details

Test Description

Test Issue Links (1)

tests

ROB-11 As a user, I can login the web application **IN PROGRESS**

Test Details

Test Type: Generic
Definition: Login Tests.Valid Login.Valid Login

Results

| Context | Output | Duration | Status |
|-----------------------------|--------|-----------|-------------|
| Open Browser To Login Page | - | 3 sec | PASS |
| Input Username | - | 24.000 ms | PASS |
| Input Password | - | 22.000 ms | PASS |
| Submit Credentials | - | 46.000 ms | PASS |
| Welcome Page Should Be Open | - | 7.000 ms | PASS |

If adopting BDD frameworks, such as Cucumber, we can also see detailed information about the Gherkin statements and deep-dive into the automation logs or evidence (e.g., screenshots) taken by the automation.

Calculator / Test Execution: CALC-7938 / Test: CALC-7936

simple integer multiplication

Import Execution ResultsExport to CucumberReturn to Test ExecutionPre

Results

| Context | Duration | Status |
|--|----------|--------|
| - | 2.167 ms | FAIL |
| Steps | | |
| Given I have entered 3 into the calculator | 0.092 ms | PASS |
| And I have entered 0 into the calculator | 0.706 ms | PASS |
| When I press multiply | 0.047 ms | PASS |
| Then the result should be 0 on the screen | 1.322 ms | FAIL |

```
java.lang.AssertionError: expected:<0> but was:<3>
    at org.junit.Assert.fail(Assert.java:89)
    at org.junit.Assert.failNotEquals(Assert.java:835)
    at org.junit.Assert.assertEquals(Assert.java:120)
    at org.junit.Assert.assertEquals(Assert.java:146)
    at calculator.StepDefinitions.the_result_should_be_on_the_screen(StepDefinitions.java:36)
    at *.the result should be 0 on the screen(file:///Users/smsf/exps/cucumber-java-calc/features/2_CALC-7935.feature:11)
```

Test Details

Test Type: Cucumber

Scenario Type: Scenario

Scenario:

1 Given I have 42 cukes in my belly

2 When I wait 1 hour

3 Then my belly should growl

Results

| Context | Duration | Status |
|-----------------------------------|--------------|--------|
| - | 200 millicec | PASS |
| Hooks | | |
| Before Cukes.setUp() | 0 millicec | PASS |
| After Cukes.tearDown() | | PASS |
| Background | | |
| Given buy a few cukes | | PASS |
| Steps | | |
| Given I have 42 cukes in my belly | | PASS |
| When I wait 1 hour | | PASS |
| Then my belly should growl | | PASS |

evidence_step_30_0.png

evidence_step_30_1.jpg

evidence_step_30_2.txt

evidence_step_30_3.html

evidence_step_30_4.xml

Xray also supports exploratory testing using a desktop app ([Xray Exploratory App](#)) that can integrate with Xray to bring the best of both worlds: have exploratory testing evidence tracked in Jira, be accessible by the team, and reflect on the related requirements.

Execution Status **FAILED**



Started On: 4/Dec/2019 04:30 PM

Finished On: 4/Dec/2019 06:43 PM

Assignee:

Unassigned

Executed By:

[Nelson Pereira](#)

Test Environments: -

Versions: -

Revision: -

Comment

Session Started • 18h36 | 04-12-2019
Duration 00:01:02

00:00:52 • Note [Idea]
A great note about this

Execution Defects (1)

HX-204 my defect documented

Execution Evidence (2)

xrayVideo_2019-... 266.7 kB 4/Dec/2019 06:41 PM

Test Session-D-... 2.1 MB 4/Dec/2019 06:43 PM

Execution Details

Learn more:

- [Execute Tests](#)
- [Vast collection of test automation tutorials showcasing integrations with Xray](#)
- [Performing exploratory testing and integrating it with Xray](#)

Ensure data persistence, history and change tracking

In order to enable auditing and facilitate diagnosis, data must be stored, and changes, whenever applicable, need to be clearly identified.

In Xray, all data is persisted and can be easily tracked using a historical timeline.

All changes on Jira issue based entities (i.e. Test, Pre-Condition, Test Set, Test Plan, Test Execution, Sub-Test Execution) are tracked similarly to other issue types in Jira.

One of those examples include Tests themselves.

Activity

AllCommentsWork LogHistoryActivity

Administrator created issue - 24/Jun/19 10:20 AM

Administrator made changes - 24/Jun/19 10:20 AM

| Field | Original Value | New Value |
|-------------------|----------------|--|
| Component/s | | Account [10900] |
| Priority | Trivial [5] | Major [3] |
| Manual Test Steps | | [Step: Step] [Data: Data] [Expected Result: Expected Result] |
| Manual Test Steps | | [Step: Open the Change Password screen by selecting option "My Profile > Password"] [Data:] [Expected Result:] |











Test Runs also have an activity section where its changes are tracked.








Dataset

Return to Test Execution

Previous

| | | | | |
|---|---|---|--|---|
| Actual Result ▾ | Comment  + | Defects  (0) + | Evidence  (0) + | Step State  PASS  |
| 4 Action Close error message and fill again the password fields with data | | Data Current Password: P4ssw0rd New Password: P4ssw0rd Confirm New Password: P4ssw0rd | | Expected Result Information message: "Password successfully changed" |
| Actual Result ▾ | Comment  + | Defects  (0) + | Evidence  (0) + | Step State  PASS  |

Activity

-  xadmin changed status from **EXECUTING** to **PASS**.
Just now
-  xadmin edited **STEP 4**
• Changed field "Status" from **TODO** to **PASS**
Just now
-  xadmin edited **STEP 3**
• Changed field "Status" from **TODO** to **PASS**
Just now
-  xadmin edited **STEP 2**
• Changed field "Status" from **TODO** to **PASS**
Just now
-  xadmin changed status from **TODO** to **EXECUTING**.
Just now

Learn more:

- [Test Runs](#)

Ensure no data tampering

Historical results (i.e. Test Run details) can't be modified. The same applies to past changes made on Jira issue based entities (i.e. Test, Pre-Condition, Test Set, Test Plan, Test Execution, Sub-Test Execution).

It is also possible to make Xray issue-based entities read-only. Besides, test specification cannot be modified during execution of the related test.

Export data in human-readable formats and automate data snapshots

All core entities, including Test Runs (i.e. results), can be exported to PDF, Word, or Excel documents, using fully customizable documents in terms of layout.

This is a way to obtain a formal, readable, copy of the relevant test data, no matter if it's related to test specification or execution.

Two options exist to achieve this: either using the built-in [Document Generator](#) capabilities, or by using the more complete and general purpose [Xporter App](#).

You can start by defining a template, or downloading and customizing an existing one from the [template store](#), with all the data to be embedded, logos and other information (e.g., legal disclaimers)



Xray - Document Generator

Template: Test Plan Advanced with Cover Page (Xray 5)

Output Format: DOCX

Buttons: Export, Close

Test Plan for v1.0

Details: Type: Test Plan, Priority: Trivial, Affects Version/s: None, Component/s: None, Labels: default

Tests: 18 PASS, 2 FAIL

| Key | Summary | Requirements | #Test Executions | Issue Assignee | Latest Status |
|---------|---|----------------|------------------|----------------|---------------|
| BOOK-18 | Test a visitor can change his locale | BOOK-5 | 1 | Bruno Conde | PASS |
| BOOK-17 | Test a logged in visitor can edit the default address | BOOK-1, BOOK-4 | 1 | Bruno Conde | PASS |

With Xporter it's also possible to export issues in bulk; this can be useful to export a set of Tests or Test Executions, for example.

Bulk Operation

Step 2 of 4: Choose Operation

Choose the operation you wish to perform on the selected 3 issue(s).

- ☐ Edit Issues: Edit field values of issues
- ☐ Move Issues: Move issues to new projects and issue types
- ☐ Transition Issues: Transition issues through workflow
- ☐ Delete Issues: Permanently delete issues from Jira
- ☐ Watch Issues: Watch all the selected issues. You will receive notifications when any of these issues are updated.
- ☐ Stop Watching Issues: Stop watching all the selected issues. You will no longer receive notifications when any of these issues are updated.
- ☐ Export Issues (Xporter): Export issues using Xporter

Buttons: Next, Cancel

With any of the two solutions, and just for reference, it's possible to:

- generate a custom PDF document with test cases and their specification, and the link to the related requirements (e.g., the typical "notebook of tests)
- generate a custom Word document with the current detailed results of a Test Plan, and all related iterations (i.e. Test Executions)

With Xporter it's possible to automate the generation of these documents and, for example, generate them upon a transition of a workflow, or attach to an existing Confluence page, or even send them to some email.

Administration

Search JIRA admin

ApplicationsProjectsIssuesAdd-onsUser managementSystem

Add Post Function To Transition

| Name | Description |
|---|--|
| <input type="radio"/> Assign to Current User | Assigns the issue to the current user if the current user has the 'Assignable User' permission. |
| <input type="radio"/> Assign to Lead Developer | Assigns the issue to the project/component lead developer |
| <input type="radio"/> Assign to Reporter | Assigns the issue to the reporter |
| <input type="radio"/> Create Perforce Job Function | Creates a Perforce Job (if required) after completing the workflow transition. |
| <input type="radio"/> Fire Event | Fires an event that can be processed by listeners. |
| <input type="radio"/> Generate Change History | Updates change history for an issue and stores the issue to the database. |
| <input type="radio"/> Notify HipChat | Send a notification to one or more HipChat rooms. |
| <input type="radio"/> Trigger a Webhook | If this post-function is executed, JIRA will post the issue content in JSON format to the URL specified. |
| <input type="radio"/> Update Issue Field | Updates a simple issue field to a given value. |
| <input type="radio"/> Update Issue Status | Sets issue status to the linked status of the destination workflow step. |
| <input type="radio"/> Xporter for JIRA Create Document Post Function | Create a document based on the issue fields. |
| <input checked="" type="radio"/> Xporter for JIRA Send Report Post Function | Send a report by email. |

Add

Cancel

- Learn more:
- Built-in Document Generator
 - Xporter app
 - Xporter Template Store

Risks

Handle risks with proper Risk Management and Risk-Based Testing

Handling risks is an intrinsic part of good testing and is essential to organizations that work in highly regulated environments.

Xray supports Risk-Based Testing (RBT) and allows you to define risks at different levels: project, requirement or at the test-level.

Depending on the exact need, implementation of risk management can be accomplished using Jira built-in capabilities or through an additional app.

No matter the approach, we can use risks to pick the requirements and/or the Tests that we need based on risk criteria.

Add Tests to Test Execution CALC-3702
Assignee Issue Assignee

SelectSearchJQL

Filter(s) ▾

Project

Calculator (CALC) × ▾

Test Type

▾

Choose the Test Type

Contains text

Risk Score between

2

and

16

The Risk Score Field for Risk Management

Clear

Search

☐ Issue Type
☐ Key
☐ Summary
☐ Risk Level
☐ Probability
☐ Impact
☐ Risk Score

| | | | | | | | |
|--------------------------|-------------------------------------|-----------|--|------------|---|---|------------|
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | CALC-3703 | ability to perform clear / start new operations from scratch | L5: Severe | 4 | 4 | (((16))) |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | CALC-1202 | CanAddNumbers | L3: High | 2 | 4 | ((8)) |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | CALC-1203 | CanDoStuff | L2: Medium | 1 | 4 | (4) |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | CALC-3706 | check the history of operations | L1: Low | 1 | 2 | (2) |

Showing 1 to 4 of 4 entries

FirstPrevious1NextLast

☒ Hide Tests in non-executable Statuses

Add selected
Add all (4)
Cancel

Whenever risks are defined at requirement level, we can analyze coverage for all requirements on our project and, for example, decide to make a release or not based on the status of the most critical requirements.

Overall Requirement Coverage Report [Switch report ▾](#)



Learn more:

- Performing Risk-Based Testing (RBT) with Xray
- Risk Management
- Risk-Based Testing

Mitigate risks by fostering clarity and facilitating collaboration, including on all testing activities

The best way to avoid misunderstandings is to clarify them ASAP and make sure doubts don't get lost. This is, in itself, a way of risk mitigation.

Comments and evidence (e.g., reference documentation) can be added on all Xray issue based entities (i.e. Test, Pre-Condition, Test Set, Test Plan, Test Execution, Sub-Test Execution) and also on Test Runs.

Details

Type: Test
Priority: Trivial
Affects Version/s: None
Component/s: None
Labels: default
Test Repository Path: Shopping Basket

Description

Test Details

Type: Cucumber
Scenario Type: Scenario
Scenario: Given I am on a product detail page
When I select the size/color/amount
And I click the add to basket button
Then the product is added to my shopping basket

Pre-Conditions

Test Sets

Test Plans

Test Runs

Attachments

Issue Links

Structure

Activity

All

Comments

Work Log

History

Activity

Xpand IT Admin added a comment - Just now

Bruno Conde are there any security risks we should consider in this flow?

Status: OPEN (View Workflow)
Resolution: Unresolved
Fix Version/s: 1.0

Xporter

Template: Basic Release Notes
Output format: DOCX
Export

People

Assignee: Bruno Conde
Assign to me
Reporter: Xpand IT Admin
Votes: 0
Watchers: 1 Stop watching this issue

Dates

Created: 25/Feb/19 1:39 PM
Updated: Just now

Agile

View on Board

Edit Steps

...

+

⚙

If your team decides you need one or more specific fields to embed some relevant information, then you can create custom fields on all of the previous entities, including on Test Steps and also on Test Runs, if you wish to do so.

Bookstore / BOOK-23

Test visitors can add books to their shopping basket

Edit Comment Assign More Start Progress Resolve Issue Workflow Admin

Details

Type: ☒ Test Status: **OPEN** (View Workflow)
 Priority: ☐ Trivial Resolution: Unresolved
 Affects Version/s: None Fix Version/s: 1.0
 Component/s: None
 Labels: default
 Test Repository Path: Shopping Basket

Description

Test Details

Type: Cucumber
 Scenario Type: Scenario
 Scenario:
 Given I am on a product detail page
 When I select the size/color/amount
 And I click the add to basket button
 Then the product is added to my shopping basket

Add Field

Name* Probability

Probability (Create new field)

Create field Cancel

Jira Dashboards Projects Issues Boards Structure Xporter Reports easyBI Tests Create

Search

Project settings

Summary Details Audit log Re-index project Delete project

Issue types

- Bug
- Epic
- Pre-Condition
- Story
- Sub-bug
- Sub-task
- Sub Test Execution
- Task
- Test
- Test Execution

2 more issue types

Test Run Custom Fields

Define Test Run "Custom fields" at project level

| Name | Description | Flags | Actions |
|------|-------------|-------|---------|
|------|-------------|-------|---------|

Create Test Run Field

Name* hw temperature

Description

Type Number Field

Test Types*

Required ☒ This test run field is required

Create Cancel

Learn more:

- [Advantages of using Jira issue types in Test Management](#)
- [Configuring Test Step Custom Fields](#)
- [Configuring Test Run Custom Fields](#)

Trace relationship between entities, up to the code

Xray provides full traceability between requirements, tests, their runs, and reported defects. This traceability not only provides the relation between the entities but also their status based on testing.

Traceability can be evaluated for a specific version and/or a specific environment, as its contents will depend on the testing results obtained for that specific context.

In the following screenshot, we can see an Epic that is NOK on version 1.0 even though one of the related Story issues is OK. In this example, the latest result for one of the tests (BOOK-23) that cover the Epic was a failure. It's also possible to see that a defect (BOOK-32) has been reported on that Test Run.

Traceability Report

Switch report

Scope: Version; Version: 1.0; Environment: All EnvironmentsContains text: BOOK-8How to read this report

Showing 1 of 1 entries

| Requirement | Tests | Test Runs | Defects |
|--|--|--|---|
| <div><div>BOOK-8</div><div>1.0 - NOK</div><div>Version: 1.0</div><div>As a visitor, I can manage my Shopping Basket</div></div> | <div><div>BOOK-24</div><div>Test visitors can remove books from their shopping basket</div><div>1.0 - PASS</div></div> <div><div>BOOK-25</div><div>Test a visitor can view all the books in his shopping basket</div><div>1.0 - PASS</div></div> <div><div>BOOK-23</div><div>Test visitors can add books to their shopping basket</div><div>1.0 - FAIL</div></div> | <div><div>PASS</div><div>BOOK-31</div><div>View Details</div><div>Version: 1.0</div><div>Finished On:25/Feb/19 2:07 PM</div><div>Executed By:xadmin</div><div>Tests environments:-</div><div>Revision:65d64uyg7t6r</div></div> <div><div>PASS</div><div>BOOK-31</div><div>View Details</div><div>Version: 1.0</div><div>Finished On:25/Feb/19 2:07 PM</div><div>Executed By:xadmin</div><div>Tests environments:-</div><div>Revision:65d64uyg7t6r</div></div> <div><div>FAIL</div><div>BOOK-31</div><div>View Details</div><div>Version: 1.0</div><div>Finished On:25/Feb/19 2:10 PM</div><div>Executed By:xadmin</div><div>Tests environments:-</div><div>Revision:65d64uyg7t6r</div></div> | <div><div>BOOK-32</div><div>Error when adding Book to shopping basket</div></div> |
| <div><div>BOOK-11</div><div>1.0 - OK</div><div>Version: 1.0</div><div>As a visitor, I can view all the books in my shopping basket</div></div> | <div><div>BOOK-25</div><div>Test a visitor can view all the books in his shopping basket</div><div>1.0 - PASS</div></div> | <div><div>PASS</div><div>BOOK-31</div><div>View Details</div><div>Version: 1.0</div><div>Finished On:25/Feb/19 2:07 PM</div><div>Executed By:xadmin</div><div>Tests environments:-</div><div>Revision:65d64uyg7t6r</div></div> | |

This helps auditing and diagnosing, knowing exactly what is impacting on what.

In Jira we can even look at the code of the requirements (e.g., Epic or Story issues) to facilitate analysis. The same can also be applied to automated tests that will have a corresponding Test issue that we can reference on the code commits. This helps analyze if a potential reported bug is on the features or even on the test automation code.

14: 2 pull requests

| ID | Title | Status | Author | Reviewer | Updated |
|-------|-------------|--------|--------|----------|------------|
| #4149 | ... | MERGED | | | 3 days ago |
| #4136 | Feature/... | MERGED | | | 4 days ago |

Start watching this issue

11/Mar/21 7:44 PM
49 minutes ago

0m

0m

1w 2d 4h 43m

Updated 3 days ago
Latest 3 days ago
Updated 3 days ago
Latest Yesterday

Development

2 branches

9 commits

2 pull requests

MERGED

Updated 3 days ago

Latest 3 days ago

Updated 3 days ago

Facilitated diagnosis and Root-Cause Analysis

Root Cause Analysis (RCA) involves answering questions like these:

- **What** is the root **cause** of a problem?
- **When** was a problem introduced?
- **Why** was the problem introduced?
- **Where** was the problem introduced at?
- **How** can it be prevented to happen in the future?

Xray helps answer these questions in multiple ways, using all the capabilities mentioned earlier.

To find when a problem was introduced, for example, we can look at coverage information which can be used to identify the status of the requirement on each version. We can then use the Traceability Report or the Test Runs List Report to see in which revision it was reported, comments and other testing evidence left.

Learn more:

- [\[Xray\] Xray's features for handling Root Cause Analysis](#)

Learning

Facilitated training resources to improve knowledge and ensure a seamless adoption

Understanding a tool, how it works, its recommend use, possible configurations and extensions/integrations is essential to make the most out of it and fine tune it to the team and overall process needs.

Besides the extensive user guide documentation available online, there are also many courses on Xray Academy to understand not only Xray essentials but also more specific/advanced topics, such as test automation.

Learn more:

- [Xray Academy](#)
- [Xray server/datacenter user guide](#)

Standards and regulatory requirements

Standards and regulation that apply to organizations depend on the sector(s) they are working at and the type of products being developed and maintained.

For Medical Device Regulation (MDR), for example, these are some of the potential standards to consider:

- ISO 13485
- IEC 62304:2006
- ISO 14791:2019
- US FDA CFR Part 11