

# Documenting Release Requirements and Use Cases in JIRA

## Introduction

This document describes documenting requirements and use cases, as well as integration testing, using Jira.

### UPDATE Dec 1, 2020

The "Req or Use Case" field has been updated to accommodate changes to the release process, beginning with the Honolulu release.

#### 1. Field name change

*WAS: "Req or Use Case"*

*IS: "Requirement Type"*

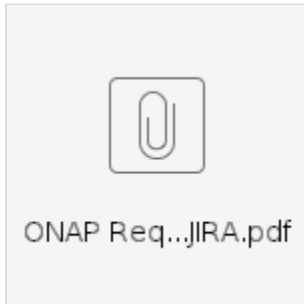
#### 2. Deprecated selections. The following selections will no longer be used:

- Requirement
- Non-Functional Requirement

#### 3. New selections. The following selections have been added. You may find definitions [here](#).

- Best Practice (new code only)
- Best Practice (global - all code)
- Specification
- Feature

### UPDATE May 1, 2020




After using the requirements JIRA for the Frankfurt release, I've made some changes to improve the process, which we will begin using with the Guilin release:

1. Removed sub-tasks for TSC milestone approvals. Now these are custom fields within the epic.
  - a. This also simplifies the cloning process, since users no longer need to clone the subtasks, which created some confusion in Frankfurt.
2. Added fields for:
  - a. Requirement or Use Case
  - b. Proof of Concept (PoC)
  - c. t-shirt size,
  - d. milestone scorecards,
  - e. milestone TSC approvals,
  - f. arch subcommittee review status (not yet performed, not required, GO, NO GO)
  - g. scope status (original scope, reduced scope, de-scoped)
3. Added a template in the description to capture:
  - a. basic description of use case or requirement
  - b. Link to HLD/LLD
  - c. Dependency Relationships with Other Projects
  - d. Project Impact
  - e. Support Status for each Affected Project
  - f. Integration Leads
  - g. Company Engagement
4. ALL data contained in the release requirements table is now captured in the JIRA issue

- [Introduction](#)
  - [UPDATE Dec 1, 2020](#)
  - [UPDATE May 1, 2020](#)
- [Process](#)
  - [Requirement Creation](#)
  - [Requirements Updates at Release Milestones](#)
  - [Integration Test Task Creation](#)
    - [If Your Integration Test Task is Blocked](#)
  - [Continuing a Requirement From One Release to the Next Release](#)
- [Structure and Organization of Related Issues](#)
- [Frequently Asked Questions \(FAQ\)](#)
  - [Q: Is the Release Requirements \(REQ\) JIRA issue intended to replace issues documented in other JIRA projects?](#)
  - [Q: Should I use REQ to document technical content?](#)
  - [Q: May I have more than one release requirement JIRA associated with a requirement in the release requirements table in the wiki?](#)
  - [Q: May I have the same technical issue, such as a user story, associated with more than one requirements JIRA issues?](#)
  - [Q: May I have a release requirement that is not associated with a user story.](#)
  - [Q: Should I make a requirements JIRA issue a dependency of another issue?](#)
- [For More Information](#)

# Process

## Requirement Creation

1. Navigate to 
2. Select More==>Clone++
3. Clone the issue in the same project (i.e., REQ).
4. Modify the "epic name" and "summary" fields for your requirement. These may be the same, if you wish.
5. Click the "Create" button.
6. Click the "Edit" button.
7. The "fixVersion" field should already be set, but if not, set it to the release for which the requirement is intended (e.g. "Guilin Release")
8. Set the following fields, if known. **Note: If not known, or not yet determined, wait until the information is known to set the field.**
  - a. Assignee - this should be one of the requirements owners, if more than one. Use the template in the description field to document all of the requirements owners.
  - b. Requirement Type - select the appropriate type. Avoid the selections that say "DEPRECATED". Definitions may be found [here](#).
  - c. PoC - check this box if the requirement or use case is a Proof of Concept (PoC)
  - d. TSC priority (0 - 4). **Note: this field is set by the TSC. Please do not change the field unless you have received direction from the TSC.**
  - e. Architectural Review Status (defaults to "not yet done")
    - i. Note: architectural review status is determined by the arch subcommittee. **Do not set this field until you get feedback from the subcommittee.**
    - ii. Upon completing arch review, please link the ONAPARC JIRA as a blocking issue ("is blocked by") for the REQ JIRA.
  - f. T-Shirt Size (defaults to M)
9. If it has not yet been done, fill in the template contained in the "description" field.
10. You may do this now, or later: Create a "story" issue in a project or committee JIRA project and link it to the requirement issue by setting the "linked issues" field to "blocked by" in the requirements issue.
  - a. All of your technical issues related to the requirement should be linked to the story.
11. Click "Update".

## Requirements Updates at Release Milestones

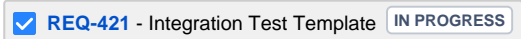
Note: the fields mentioned below are not initially visible in JIRA unless in edit mode. Once the field is set, it will appear on the form after it is updated.

Note: Each update to the requirement should be accompanied by a comment that explains the update, as well as any clarifying information about status.

1. By M1, all of the fields mentioned above under "Issue Creation" should be known and entered in the issue. If not, the requirement should not be approved for M1.
2. Enter the scorecard for the milestone as well as a comment that explains the scorecard and the current status. **A scorecard without an accompanying comment is not acceptable.**
3. The release manager or the TSC chair will enter the TSC decision on approval of the requirement for the milestone.
  - a. If the TSC has questions about the status of the requirement, then a question will be entered as a comment to the issue. The requirement owner must respond to the question by adding another comment to the issue.
4. If the requirement is reduced in scope, or de-scoped, then update the "Scope status" field (defaults to "Original scope").
5. If the requirement is changed to a Proof of Concept (PoC), then check the "PoC" box.
6. Once the requirement is either completed, or de-scoped, then mark the issue "Closed". Add a comment explaining why the issue is being closed.

## Integration Test Task Creation

Beginning with the Guilin release, we will replace the integration test status table with a an embedded Jira table, similar to what was done with requirements and use cases.

1. Navigate to 
2. Select More==>Clone++
3. Clone the issue in the same project (i.e., REQ).
4. Modify the "summary" field.
5. Set the 'Assignee' field.
6. Set the 'Fix Version' field to the current, in-process release name.
7. Set the 'Epic Link' field. This should be the REQ issue (Epic) that represents the requirement.
8. Click the "Create" button.
9. The label, "integration\_test\_task" should already exist, but if not, add it to the "labels" field.

10. Click the "Edit" button.
11. Set the following fields, if known. **Note: If not known, or not yet determined, wait until the information is known to set the field.**
  - a. Integration Test Plan Status
    - Not started
    - In process
    - In review
    - Completed
  - b. Integration Test Status
    - Not started
    - In process (< 50%)
    - In process (>= 50%)
    - Completed (*also set Integration Test Time to Complete to empty*)
    - BLOCKED (*see instructions below*)
  - c. Integration Test Time to Complete
    - Please keep this field updated with the expected time to complete testing.
    - When Integration Test Status is "Completed", set this field to empty
12. If it has not yet been done, fill in the template contained in the "description" field.
13. Click "Update".

## If Your Integration Test Task is Blocked

1. Open the Jira (REQ) that documents the integration test that is blocked and select edit.
2. Locate the "Integration Test Status" field and select "BLOCKED". Save your changes.
3. Create a new issue to document the reason for being blocked. For example, if the test is being blocked because of an issue in SO, then create a new issue in the SO project (SO-XYZ).
  - a. Note: if the blocking issue is already documented in Jira, simply link that issue. You do not need to create a new issue.
4. Link the new issue to your integration test issue (Blocked by).
5. **IMPORTANT:** Add a comment to your integration test issue that explains the problem and include a reference to the new issue that you created.

## Continuing a Requirement From One Release to the Next Release

Sometimes a requirement may need two or more releases to be completed, so how should this be tracked in Jira? In order to preserve the history of the requirement, please create a new requirement for each release, as follows:

### For the Current Issue:

1. Add a comment:
  - What was completed
  - What remains to be done
  - Assert that the requirement will be continued in the next release
  - A reference (key) to the new requirement, i.e., REQ-xxxx.
2. Change the "scope status" to "reduced scope"
3. Update the Jira status as "Done"

### For the New Issue:

1. Create a new issue with identical summary as the previous issue.
2. Set fixversion to the target release version (e.g., Honolulu Release).
3. In the "Issue Links" field, add a link to the previous issue.
4. Add a comment indicating that this issue is a continuation of a previous issue (add Jira reference)

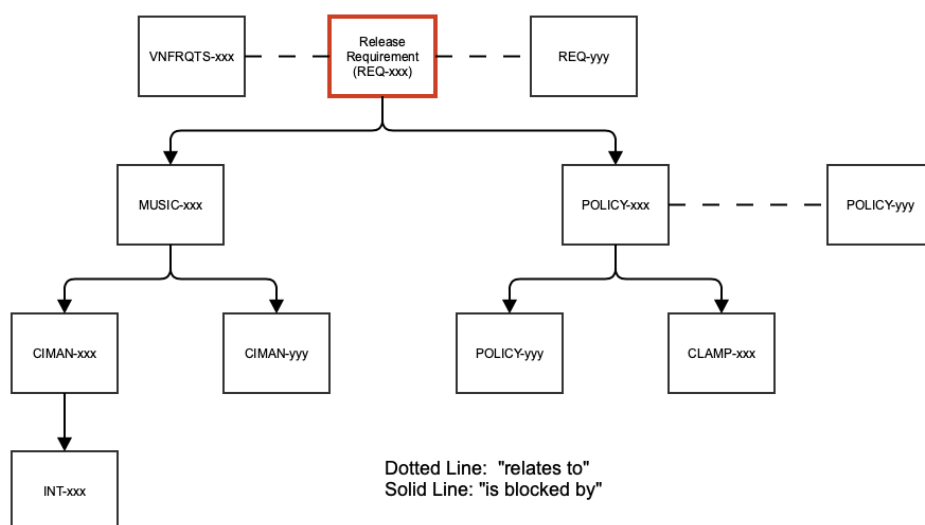
## Structure and Organization of Related Issues

The purpose of the Release Requirements (REQ) JIRA project is to create a single point from which all issues related to a particular release requirement may be found. This association is nicely accomplished using the JIRA linking capability. We will use two links, in particular: 1) "is blocked by"; and 2) "is related to". Notice that we do not use "blocks". This is because we want the REQ issue to be at the root of a network of related issues. The REQ issue served an administrative function by creating a single connecting point for the network, while all of the other issues fill in the user story, features, tasks, and bugs.

The diagram below shows an example of what this might look like. It shows a release requirement (red box) with a chain of dependent issues. The dependent issues are where the technical content is located. However, notice that it also shows some "relates to" relationships (dotted lines). These may include relationships to technical issues in other projects, such as a user story, but they may also include relationships to other release requirements. In any case, we are able to see all of the detail of the release requirement by beginning with the issue in the red box and following the links.

For a real example of how the issues should be organized relative to a release requirements JIRA issue,

see [REQ-43](#) [DONE](#) .



## Frequently Asked Questions (FAQ)

### Q: Is the Release Requirements (REQ) JIRA issue intended to replace issues documented in other JIRA projects?

A: No! REQ served an administrative purpose by linking all of the technical issues (user story, features, tasks, etc.) that may be spread out over multiple JIRA projects to one location. So, REQ does not replace your technical JIRA issues, but instead provides a single link to all of those issues (see diagram).

### Q: Should I use REQ to document technical content?

A: No. As mentioned above, REQ serves an administrative function, not a technical function. It links all of the technical issues for a requirement to a single location. Therefore, do not use REQ for technical content, but instead link the issues with technical content to REQ.

### Q: May I have more than one release requirement JIRA associated with a requirement in the release requirements table in the wiki?

A: No – there should be a 1:1 relationship between the requirements in the release requirements wiki table and the release requirements JIRA. If you have more than one JIRA per requirement that probably means that your requirement is too vague, or too broadly stated.

### Q: May I have the same technical issue, such as a user story, associated with more than one requirements JIRA issues?

Yes. This is especially true if the issue is a user story, because a user story will likely have multiple requirements associated with it

### Q: May I have a release requirement that is not associated with a user story.

Yes.

### Q: Should I make a requirements JIRA issue a dependency of another issue?

No. As mentioned above, requirements JIRA issues should be at the root of the network. If you want to associate another issue with a requirements issue, that is not a dependency, try using the "is related to" link.

## For More Information

Please contact [David McBride](#)

