

# Integration with Automation for Jira

- [Overview](#)
- [Usage Examples](#)
  - [Trigger a Jenkins project build from an issue](#)
    - [Jenkins configuration](#)
    - [Automation for Jira configuration](#)
  - [Trigger a Jenkins project build from a Test Plan and report the results back to it](#)
    - [Jenkins configuration](#)
    - [Automation for Jira configuration](#)
- [References](#)

## Overview

[Automation for Jira](#) app enables users to easily extend and implement automation in Jira without having to code.

This way, users can implement rules that are triggered upon some event, executed if certain condition(s) are met and, that perform certain action(s).

Rules can also be triggered manually or may be scheduled.

Since Xray uses issue types for most of its entities and since Xray provides many JQL functions that allow you to obtain testing-related information, Automation for Jira can be used with Xray in a very straightforward way.

Automation rules are available and, can be created, from the project settings, namely from the "Automation" tab.

Project settings

Automation

Name*	Owner	Project	Enabled
Trigger Jenkins job	Administrator	Calculator (CALC)	✓
Trigger Jenkins job and link to Test Plan	Administrator	Calculator (CALC)	✓



### Please note

The following examples are provided as-is, no warranties attached; use them carefully.

Please feel free to adapt them to your needs.

Note: We don't provide support for Automaton for Jira; if you have doubts concerning its usage, please contact [Automation for Jira's support](#).

## Usage Examples

### Trigger a Jenkins project build from an issue

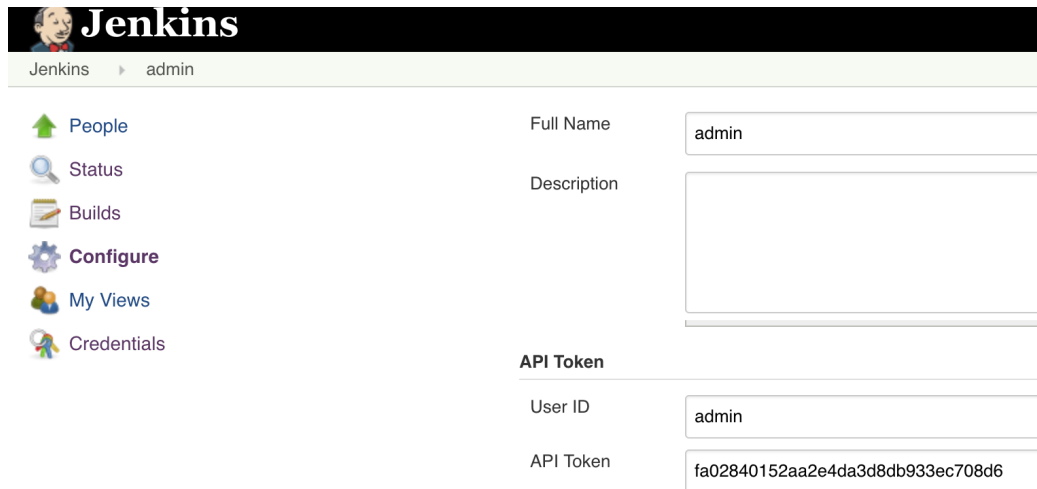
In this very simple scenario, we'll implement a rule, triggered manually, that will trigger a Jenkins project/job. The action will be available from within the "More" menu, in all issues of the selected project.

We're assuming that:

- you just want to trigger a CI job, period; this job may be totally unrelated to the issue from where you triggered it
- what the CI job will do, including if it will report the results back to Xray or not, is not relevant

## Jenkins configuration

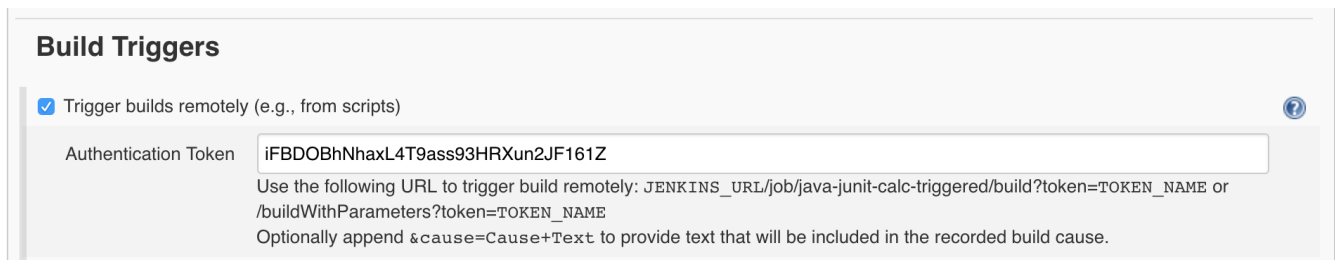
In Jenkins, we need to generate an API token for some user, which can be done from the profile settings page.



The image shows the Jenkins Admin page for the 'admin' user. On the left is a sidebar with navigation links: People, Status, Builds, Configure, My Views, and Credentials. The main content area shows the user's profile information. The 'Full Name' field is 'admin'. The 'Description' field is empty. The 'API Token' section shows the 'User ID' as 'admin' and the 'API Token' as 'fa02840152aa2e4da3d8db933ec708d6'.

Full Name	admin
Description	
<b>API Token</b>	
User ID	admin
API Token	fa02840152aa2e4da3d8db933ec708d6

At the project level, we need to enable remote build triggers, so we can obtain an "authentication token" to be used in the HTTP request afterwards.



The image shows the 'Build Triggers' configuration page in Jenkins. The 'Trigger builds remotely (e.g., from scripts)' checkbox is checked. The 'Authentication Token' field contains the value 'iFBDOBHnaxL4T9ass93HRXun2JF161Z'. Below the field, there is a text box explaining how to use the token to trigger a build remotely, including the URL format and optional parameters.

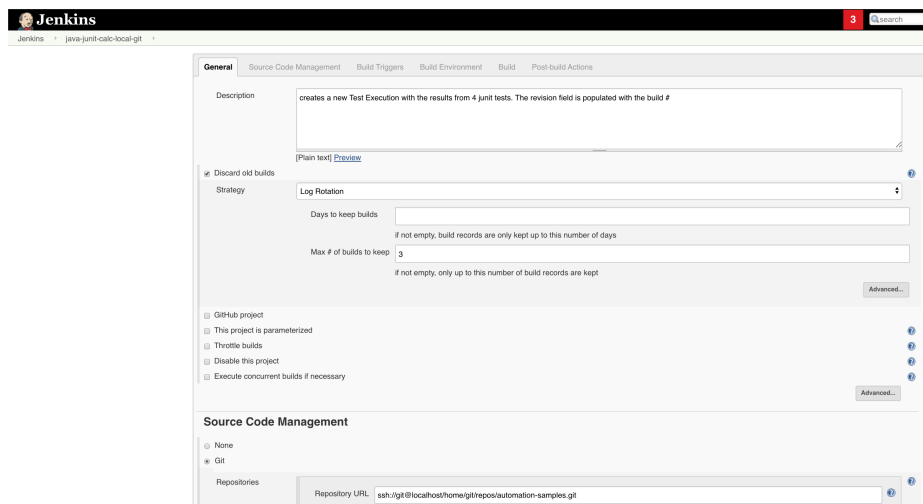
☒ Trigger builds remotely (e.g., from scripts)

Authentication Token:

Use the following URL to trigger build remotely: `JENKINS_URL/job/java-junit-calc-triggered/build?token=TOKEN_NAME` or `/buildWithParameters?token=TOKEN_NAME`

Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

The project itself is a normal one, without parameters.



The image shows the Jenkins project configuration page for a project named 'java-junit-calc-local-git'. The 'General' tab is selected. The 'Description' field contains the text 'creates a new Test Execution with the results from 4 junit tests. The revision field is populated with the build #'. The 'Discard old builds' checkbox is checked. The 'Strategy' dropdown is set to 'Log Rotation'. The 'Days to keep builds' field is empty. The 'Max # of builds to keep' field is set to '3'. The 'Source Code Management' section shows 'Git' selected as the provider. The 'Repository URL' field contains the value 'ssh://git@localhost/home/git/repos/automation-samples.git'.

**General** | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

Description: creates a new Test Execution with the results from 4 junit tests. The revision field is populated with the build #

☒ Discard old builds

Strategy: Log Rotation

Days to keep builds:

Max # of builds to keep:

Source Code Management: ☒ Git

Repository URL:

## Automation for Jira configuration

1. create a new rule and define the "When" (i.e. when it to should be triggered ), to be "Manually triggered"

Automation DRAFT Return to

Trigger Jenkins job

**Rule details**

**Name\*** Trigger Jenkins job

**Description** Trigger Jenkins job "java-junit-calc-local-git"

**Projects** Calculator (CALC)  
Projects can only be modified in the global administration.

**Enabled** ☒

**Allow rule trigger** ☐ Check to allow other rule actions to trigger this rule. Only enable this if you need this rule to execute in response to another rule.

**Notify on error** Don't notify

**Created** 3 hours ago

**Owner** Administrator  
The owner will receive emails when the rule fails.

**Updated** 3 hours ago

**Actor** Administrator  
Actions defined in this rule will be performed by the user selected as the actor.

**Save** **Cancel**

**When: Manually triggered**  
All logged in users can run rule.

**Then: Send webhook**  
POST  
http://192.168.56.102:8081/job/java-junit-calc-local-git/build?token=iFBDOBHhaxL4T9ass93HRXun2JF161Z

**Add component**

2. define an action (i.e. the "Then") as "Send webhook" and configure it as follows

Automation ENABLED

Trigger Jenkins job

**Rule details**

**Audit log**

**When: Manually triggered**  
All logged in users can run rule.

**Then: Send webhook**  
POST  
http://192.168.56.102:8081/job/java-junit-calc-local-git/build?token=iFBDOBHhaxL4T9ass93HRXun2JF161Z

**Add component**

**Send webhook**

This action will send a HTTP POST to the url specified below:

**Webhook URL\***  
http://192.168.56.102:8081/job/java-junit-calc-local-git/build?token=iFBDOBHhaxL4T9ass93HRXun2JF161Z

**Headers (optional)**

**Content-Type** application/json

**Authorization** Basic YWRtaW46YWRtaW4=

**Add**

**HTTP method**  
POST

**Webhook body**  
Empty

**Save** **Cancel**

- the Webhook URL provided above follows this syntax:
  - <jenkins\_base\_url>/job/<name\_of\_jenkins\_project\_job>/build?token=<token>
- besides the "Content-Type" header that should be "application/json", define also an "Authorization" header having the value "Basic <auth>", where the base64 encoded <auth> can be [generated](#) using your Jenkins API credentials

After publishing the rule, you can go to the screen of an issue and trigger the Jenkins project/job.

The screenshot shows the JIRA interface for a project named 'Calculator' with issue ID 'CALC-3214'. The issue title is 'all my sum related tests for v3.0'. The issue type is 'Test Plan' with a priority of 'Major'. The 'More' dropdown menu is open, showing various actions. The 'Trigger Jenkins job' option is highlighted at the bottom of the menu. The 'Overall Execution Status' shows 7 tests passed.

## Trigger a Jenkins project build from a Test Plan and report the results back to it

In this simple scenario, we'll implement a rule, triggered manually, that will trigger a Jenkins project/job. The action will be available from within the "More" menu, for all Test Plan issues of the selected project.

We're assuming that:

- you just want to trigger a CI job, period; this job may be totally unrelated to the issue from where you triggered it
- the results will be submitted back to Xray, if the project is configured to do so in Jenkins

## Jenkins configuration

In Jenkins, we need to generate an API token for some user, which can be done from the profile settings page.

The screenshot shows the Jenkins 'admin' profile settings page. The 'Full Name' field is set to 'admin'. The 'Description' field is empty. The 'API Token' section shows the 'User ID' as 'admin' and the 'API Token' as 'fa02840152aa2e4da3d8db933ec708d6'.

At the project level, we need to enable remote build triggers, so we can obtain an "authentication token" to be used in the HTTP request afterwards.

## Build Triggers

☒ Trigger builds remotely (e.g., from scripts)

Authentication Token

iFBDOBhNhaxL4T9ass93HRXun2JF161Z

Use the following URL to trigger build remotely: `JENKINS_URL/job/java-junit-calc-triggered/build?token=TOKEN_NAME` or `/buildWithParameters?token=TOKEN_NAME`

Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

The project itself is a normal one; the only thing relevant to mention is that this project is a parameterized one, so it receives TESTPLAN, that in our case will be coming from Jira.

**Jenkins** 3 Search

Jenkins > java-junit-calc-local-git-report-to-testplan

**General** Source Code Management Build Triggers Build Environment Build Post-build Actions

Description: creates a new Test Execution with the results from 4 junit tests. The revision field is populated with the build #

[Plain text] [Preview](#)

☒ Discard old builds

Strategy: Log Rotation

Days to keep builds: 3

if not empty, build records are only kept up to this number of days

Max # of builds to keep: 3

if not empty, only up to this number of build records are kept

[Advanced...](#)

☐ GitHub project

☒ This project is parameterized

**String Parameter**

Name: TESTPLAN

Default Value:

Description:

[Plain text] [Preview](#)

☒ Trim the string

[Add Parameter](#)

## Automation for Jira configuration

1. create a new rule and define the "When" (i.e. when it to should be triggered ), to be "Manually triggered"

## Automation

ENABLED

### Trigger Jenkins job and link to Test Plan

Rule details

Audit log

When: Manually triggered

All logged in users can run rule.

If: Compare two values

Checks if:  
{{issue.issue.type.name}} equals Test Plan

Then: Send webhook

POST  
http://192.168.56.102:8081/job/java-junit-calc-local-git-report-to-testplan/buildWithParameters?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z&TESTPLAN={{issue.key}}

Add component

### Rule details

Name\* Trigger Jenkins job and link to Test Plan

Description Trigger Jenkins job "java-junit-calc"

Projects Calculator (CALC)

Projects can only be modified in the global administration.

Enabled ☒

Allow rule trigger ☐ Check to allow other rule actions to trigger this rule. Only enable this if you need this rule to execute in response to another rule.

Notify on error Don't notify

Created 3 hours ago

Owner Administrator

The owner will receive emails when the rule fails.

Updated 3 hours ago

Actor Administrator

Actions defined in this rule will be performed by the user selected as the actor.

Save Cancel

2. define the condition so that this rule can only be executed from Test Plan issue

## Automation

ENABLED

### Trigger Jenkins job and link to Test Plan

Rule details

Audit log

When: Manually triggered

All logged in users can run rule.

If: Compare two values

Checks if:  
{{issue.issue.type.name}} equals Test Plan

Then: Send webhook

POST  
http://192.168.56.102:8081/job/java-junit-calc-local-git-report-to-testplan/buildWithParameters?token=iFBDOBhNhaxL4T9ass93HRXun2JF161Z&TESTPLAN={{issue.key}}

Add component

### Compare condition

Compares a value to another using value substitutions and regular expressions.

First value\*

{{issue.issue.type.name}}

Condition

Equals

Second value

Test Plan

Save Cancel

> What values can I compare?

3. define an action (i.e. the "Then") as "Send webhook" and configure it as follows

## Automation

ENABLED

### Trigger Jenkins job and link to Test Plan

① Rule details

② Audit log

When: Manually triggered

All logged in users can run rule.

If: Compare two values

Checks if:

{{issue.issueType.name}} equals Test Plan

Then: Send webhook

POST

http://192.168.56.102:8081/job/java-junit-calc-local-git-report-to-testplan/buildWithParameters?token=iFBD0BhNhaxL4T9ass93HRXun2JF161Z&TESTPLAN={{issue.key}}

+ Add component

### Send webhook

This action will send a HTTP POST to the url specified below:

Webhook URL\*

s?token=iFBD0BhNhaxL4T9ass93HRXun2JF161Z&TESTPLAN={{issue.key}}

Headers (optional)

Content-Type application/json

Authorization Basic YWRtaW46YWRtaW4=

Add

HTTP method

POST

Webhook body

Empty

Save Cancel

- the Webhook URL provided above follows this syntax:
  - <jenkins\_base\_url>/job/<name\_of\_jenkins\_project\_job>/buildWithParameters?token=<token>&TESTPLAN={{issue.key}}
- besides the "Content-Type" header that should be "application/json", define also an "Authorization" header having the value "Basic <auth>", where the base64 encoded <auth> can be [generated](#) using your Jenkins API credentials

After publishing the rule, you can go to the screen of an issue and trigger the Jenkins project/job.

Calculator / CALC-3214

all my sum related tests for v3.0

EditComment

More

Stop Progress

Resolve Issue

Close Issue

Admin

Details

Type: Test Plan

Priority: Major

Affects Version/s: None

Component/s: None

Labels: None

Sprint: Sprint 1

Test Count: 7

Description

Risks/sensible areas to cover:

- addition operation in basic mode
- addition operation in scientific mode

Tests

Test Plan Board

Overall Execution Status

7 PASS

TOTAL TESTS: 7

Filter(s)

Trigger Bamboo Build w...

Trigger Jenkins Build

Trigger Jenkins build ...

Synchronize Tests from...

Assign

Log work

Agile Board

Rank to Top

Rank to Bottom

Attach files

Voters

Stop watching

Watchers

Create sub-task

Convert to sub-task

Move

Link

Clone

Labels

Delete

Trigger Jenkins job

Trigger Jenkins job an...

Trigger Jenkins job and link to Test Plan

Status: IN PROGRESS

Resolution: Unresolved

Fix Version/s: v3.0

## References

- [Automation for Jira in the Atlassian Marketplace](#)
- [Automation for Jira documentation](#)