

# Extending and integrating with Xray using 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

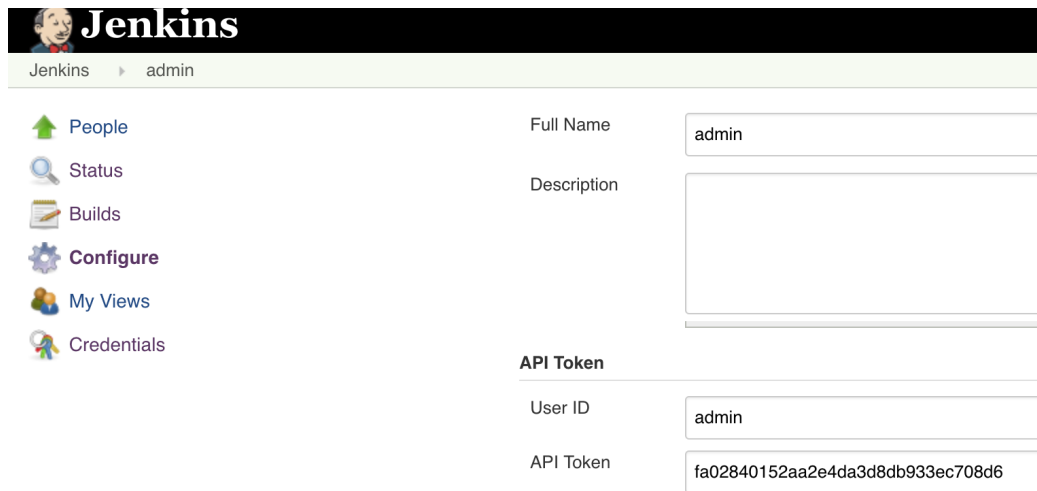
In this very simple scenario, we'll implement a rule, triggered manually, that will trigger a Jenkins project/job. The action will be available from within the "More" menu, in all issues of the selected project.

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

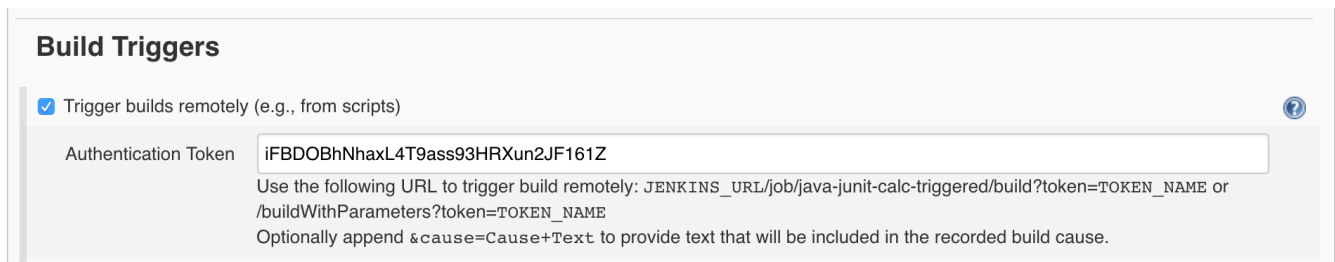
### Jenkins configuration

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



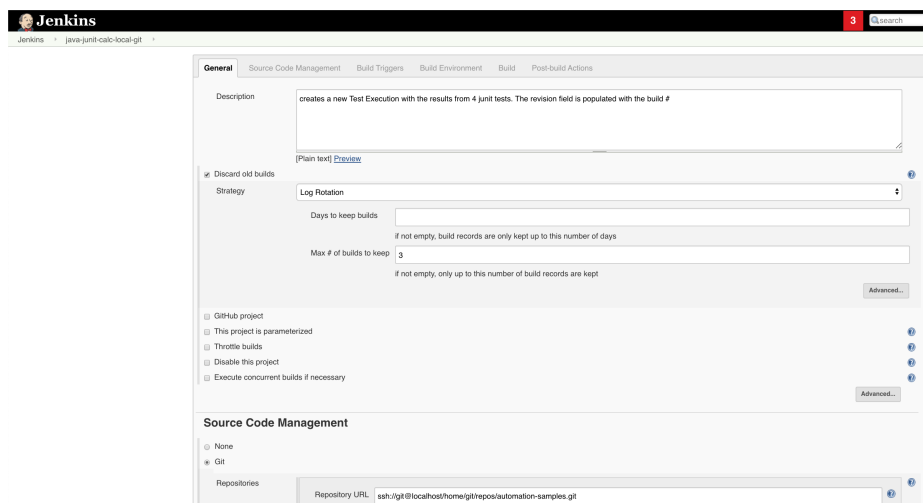
The image shows the Jenkins Admin User Configuration page for the 'admin' user. The page has a sidebar with navigation links: People, Status, Builds, Configure, My Views, and Credentials. The main content area is divided into two sections. The top section contains fields for 'Full Name' (admin) and 'Description'. The bottom section, titled 'API Token', contains fields for 'User ID' (admin) and '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 Jenkins Build Triggers configuration page. The 'Trigger builds remotely (e.g., from scripts)' checkbox is checked. Below it, the 'Authentication Token' field contains the value 'iFBDOBHnaxL4T9ass93HRXun2JF161Z'. A text box below the token field provides instructions: '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 the 'java-junit-calc-local-git' project. 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' section is expanded, showing 'Log Rotation' as the strategy, 'Days to keep builds' set to 3, and 'Max # of builds to keep' set to 3. The 'Source Code Management' section is expanded, showing 'Git' as the provider and the 'Repository URL' as 'ssh://git@localhost/home/git/repos/automation-samples.git'.

## 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

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=iFBDOBHhaxL4T9ass93HRXun  
2JF161Z

Add component

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

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=iFBDOBHhaxL4T9ass93HRXun  
2JF161Z

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, all passing.

## 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' user profile settings page. The 'API Token' section is visible, showing 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.

The screenshot shows the Jenkins configuration page for a job named 'java-junit-calc-local-git-report-to-testplan'. The 'Build Triggers' tab is selected. The 'Trigger builds remotely' checkbox is checked, and the 'Authentication Token' is 'iFBDOBhNhaxL4T9ass93HRXun2JF161Z'. Below this, a URL template is provided: 'JENKINS\_URL/job/java-junit-calc-triggered/build?token=TOKEN\_NAME or /buildWithParameters?token=TOKEN\_NAME'. A note mentions that the URL can be extended with '&cause=Cause+Text'.

The 'General' tab is also visible, showing a description: 'creates a new Test Execution with the results from 4 junit tests. The revision field is populated with the build #'. The 'Discard old builds' section is expanded, showing a 'Log Rotation' strategy with 'Days to keep builds' set to 3 and 'Max # of builds to keep' set to 3. The 'GitHub project' section is also visible, with a checkbox for 'This project is parameterized' checked. A red box highlights the 'String Parameter' configuration for 'TESTPLAN', which has a default value of '' and a description of ''.

## 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.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=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

Edit

Comment

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...

Status: IN PROGRESS

Resolution: Unresolved

Fix Version/s: v3.0

Trigger Jenkins job and link to Test Plan

## References

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