


Integration with Cprime Power Scripts, Power Custom Fields, and Power Actions

- [Overview](#)
- [Use cases](#)
 - [Create a Test Set, Test Execution or a Test Plan programmatically](#)
 - [Validate requirement before allowing to make a transition](#)
 - [Reopen/transition linked Tests to a requirement/user story](#)
 - [Trigger CI/CD builds](#)
 - [Trigger Jenkins job and report back to Test Plan](#)
 - [Jenkins configuration](#)
 - [Power Actions configuration](#)
 - [Trigger Bamboo plan and report back to Test Plan](#)
 - [Bamboo configuration](#)
 - [Power Actions configuration](#)
 - [Display progress information, as text, on Test Execution issues](#)
- [Learn more](#)

**Please note**

The following scripts and instructions are provided as-is, no warranties attached; use with care.

Please feel free to adapt them to your needs.

Note: We don't provide support for these apps; if you have doubts concerning its usage, please contact [Cprime support](#).

Overview

Cprime provides a set of Jira apps that allow the implementation of powerful custom automation rules, including [Power Scripts](#), [Power Custom Fields for Jira](#), and [Power Actions for Jira](#).

All of those come with the SIL (Simple Issue Language) engine that provides an abstraction layer over the internal of Atlassian APIs, allowing you to easily implement and maintain Jira automation-related scripts.

	Power Scripts	Power Custom Fields for Jira	Power Actions for Jira
used to...	implement automation rules	implement custom fields whose values are derived from custom logic	implement buttons/UI elements, having a custom logic
triggering	<ul style="list-style-type: none">• workflow transitions: conditions, validators, post-functions• event listeners• schedulable	<ul style="list-style-type: none">• whenever shown	N/A

Use cases

Create a Test Set, Test Execution or a Test Plan programmatically

This use case requires the Power Scripts Fields app.

Sometimes you may need to create a Test Set, a Test Execution, or a Test Plan programmatically.

The following example shows how to create a Test Execution; it can easily be adapted for other Xray issue types. You'll need to update the corresponding REST API endpoint for adding the tests to that entity.

```

//Define struct for HTTP request
struct change {
    string [] add;
    string [] remove;
}

string jiraBaseUrl = getJIRABaseUrl();
// string jiraBaseUrl = "https://yourjiraserver.example.com";
string jiraUsername = "someuser";
string jiraPassword = "somepass";

function addTestsToTestExecution(string testExecKey, string[] testList){
    //Create array/struct
    change addRequest;

    //Add data to array/struct
    addRequest.add = testList;

    string endpointUrl = #{jiraBaseUrl} + "/rest/raven/1.0/api/testexec/" + #{testExecKey} + "/test";

    HttpRequest request;
    HttpHeaders authHeader = httpBasicAuthHeader(jiraUsername, jiraPassword);
    request.headers += authHeader;
    HttpHeaders header = httpCreateHeader("Content-Type", "application/json");
    request.headers += header;

    //Post data and get response
    string result = httpPost(endpointUrl, request, addRequest);
    logPrint("DEBUG", "result: " + #{result});
    return true;
}

string projectKey = "BOOK";
string parentIssueKey = "";
string issueType = "Test Execution";
string issueSummary = "test execution created automatically from a SIL script";
string testExecKey = createIssue(projectKey, parentIssueKey, issueType, issueSummary);
logPrint("DEBUG", "On the project " + projectKey + ", issue " + testExecKey + " was created.");

string [] testList = { "BOOK-14", "BOOK-15" };
// string jql = "project = BOOK and issuetype = Test";
// string [] testList = selectIssues(jql);

addTestsToTestExecution(testExecKey, testList);

```

Please check SIL's documentation for more [info on createIssue\(\)](#) and on [httpPost\(\)](#).

Validate requirement before allowing to make a transition

This use case requires the Power Scripts Fields app.

Sometimes you may need to assure that the requirement is actually OK before transitioning it to some status, or before resolving it.

The following script validates the requirement based on the tests executed for the version assigned to the requirement issue.

You can either make the validation based on the existence of failed tests (i.e. requirement status is "NOK") or, in a more complete way by making sure all of them are passing (i.e. requirement status is "OK").

Administration

Search Jira admin

ApplicationsProjectsIssuesManage appsUser managementSystemStructure

Add Condition To Transition

Name	Description
<input type="radio"/> Block transition until approval	Condition to block issue transition if there is a pending approval.
<input type="radio"/> Code Committed Condition	Transition to execute only if code has/has not (depending on configuration) been committed as
<input type="radio"/> Hide transition from user	Condition to hide a transition from the user. The transition can only be triggered from a worklo
<input type="radio"/> No Open Reviews Condition	Transition to execute only if there are no related open Crucible reviews.
<input type="radio"/> Only Assignee Condition	Condition to allow only the assignee to execute a transition.
<input type="radio"/> Only Reporter Condition	Condition to allow only the reporter to execute a transition.
<input type="radio"/> Permission Condition	Condition to allow only users with a certain permission to execute a transition.
<input checked="" type="radio"/> SIL Condition	Workflow condition using a SIL program
<input type="radio"/> Sub-Task Blocking Condition	Condition to block parent issue transition depending on sub-task status.
<input type="radio"/> Unreviewed Code Condition	Transition to execute only if there are no unreviewed changesets related to this issue.
<input type="radio"/> User Is In Group	Condition to allow only users in a given group to execute a transition.
<input type="radio"/> User Is In Group Custom Field	Condition to allow only users in a custom field-specified group to execute a transition.
<input type="radio"/> User Is In Project Role	Condition to allow only users in a given project role to execute a transition.

AddCancel

Administration

Search Jira admin

ApplicationsProjectsIssuesManage appsUser managementSystemStruct

Add Parameters To Condition

Add required parameters to the Condition.

Choose operation

I want to*

☒ create a new script

☐ use an existing script, unmodified

Previous

Next

Cancel

Administration

Search Jira admin

ApplicationsProjectsIssuesManage appsUser managementSystemStructure

Add Parameters To Condition

Add required parameters to the Condition.

Choose operation

Specify path

File name*

verify_coverage.sil

The filename used to save to script on the disk (or in database)

Directory*

Type to search for files

silprograms

The folder where the file will be saved

Previous

Next

Cancel

```
string version = join(fixVersions,"");
string jql = "key = "+ #{key} + " and issue in requirements('OK','" + #{project} + "','"+#{version} + "')";
logPrint("DEBUG",jql);
int numberOfIssues = countIssues(jql);
if (numberOfIssues != 1) {
    return false, "Requirement Status", "Some tests need to be executed. You must assure that they pass before making the transition.";
}
return true;
```

Reopen/transition linked Tests to a requirement/user story

This use case requires the Power Scripts Fields app.

Whenever you change the specification of a requirement/user story, you most probably will need to review the Tests that you have already specified.

The following script tries to make a transition on all linked Tests to a requirement. You can hook it to a post-function on some transition of the requirement /user story.

Administration

Search Jira admin

Applications Projects Issues Manage apps User management System Structure

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"/> Notify Hipchat	Send a notification to one or more Hipchat rooms.
<input checked="" type="radio"/> SIL Post-function	Runs a SIL program as a postfunction
<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"/> Xporter Create Document	Do NOT use this post function, it is deprecated! You may use Xporter Multi-Action.
<input type="radio"/> Xporter Multi-Action	The Xporter post function will allow users to Create a document based on the issue fields and send a report by email.
<input type="radio"/> Xporter Send Report	Do NOT use this post function, it is deprecated! You may use Xporter Multi-Action.

Add Cancel

Administration

Search Jira admin

Applications Projects Issues Manage apps User management System Structure

Add Parameters To Function

Add required parameters to the Function.

Choose operation

Specify path

I want to ☒ create a new script
☐ use an existing script, unmodified

Previous Next Cancel

Administration

Search Jira admin

Applications Projects Issues Manage apps User management System Structure

Add Parameters To Function

Add required parameters to the Function.

Choose operation

Specify path

File name* reopened_linked_tests.sil

The filename used to save to script on the disk (or in database)

Directory*

silprograms

The folder where the file will be saved

Previous Next Cancel

Please check SIL's documentation for [more info on autotransition\(\)](#).

```
string transition = "Reopen Issue"
string jql = "issue in requirementTests('" + #{key} + "')";
string [] keys = selectIssues(jql);
for(string testKey in keys) {
    logPrint("DEBUG","test_key: " + #{testKey});
    autotransition(transition, testKey);
}
```

Trigger CI/CD builds

The following examples show how you can implement some actions, using Power Actions, to trigger a build/job in a CI/CD tool such as Jenkins, Bamboo, or others.

There are some common steps:

1. define some global variables, if you want, to have the details about the CI/CD tool; this can be done using SIL Manager by creating a properties file named `sil.properties`

Administration Search Jira admin

Applications Projects Issues **Manage apps** User management System Structure

ATLASSIAN MARKETPLACE
Find new apps
Manage apps

TEAMWORKX PUSH AND PULL
FAVORITES
Push Service
Dashboard Order

EXPORTER
Global Settings
Project Settings

SIL Manager

View Selection Refresh

Select a folder to search

- silprograms
 - examples
 - bamboo_operations.sil
 - create_testexecution.sil
 - reopened_linked_tests.sil
 - sil.properties**
 - verify_coverage.sil

Check Save Run

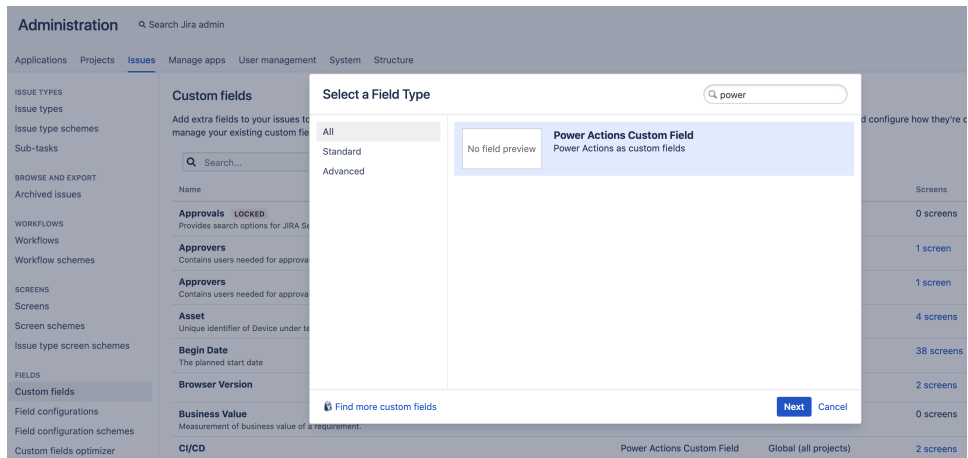
- 1 JENKINS_BASE_URL=http://myjenkins.example.com
- 2 JENKINS_USERNAME=admin
- 3 JENKINS_PASSWORD=fa02840152aa2e4da3d8db933ec708d6
- 4 JENKINS_TOKEN=iFBDOBhNhaxL4T9ass93HRXun2JF161Z
- 5 BAMBOO_BASE_URL=http://mybamboo.example.com
- 6 BAMBOO_USERNAME=admin
- 7 BAMBOO_PASSWORD=admin

a.

b. **sil.properties example**

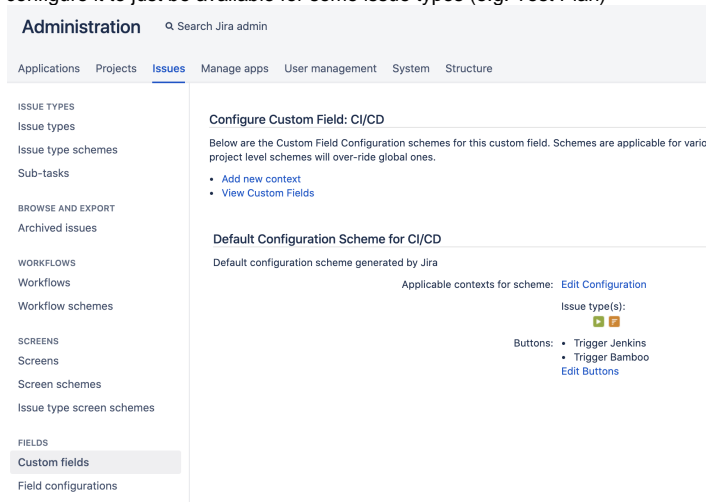
```
JENKINS_BASE_URL=http://myjenkins.example.com
JENKINS_USERNAME=admin
JENKINS_PASSWORD=fa02840152aa2e4da3d8db933ec708d6
JENKINS_TOKEN=iFBDOBhNhaxL4T9ass93HRXun2JF161Z
BAMBOO_BASE_URL=http://mybamboo.example.com
BAMBOO_USERNAME=admin
BAMBOO_PASSWORD=admin
```

2. create a Power Actions Custom Field



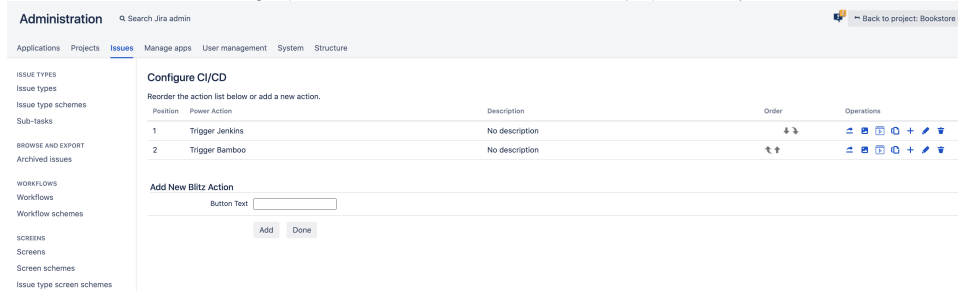
a.

3. You can configure it to just be available for some issue types (e.g. Test Plan)



a.

4. Configure the button/s and its/their logic (i.e. Condition, Screen, and Action scripts). You may create one or more buttons.



a.

b. For more info on the related scripts, please see the instructions ahead.


Trigger Jenkins job and report back to Test Plan

In this case, we're going to give users the ability to trigger an existing job/project in Jenkins.


We'll obtain the list of available jobs/projects using Jenkins REST API which we'll present to the user. Then, we trigger the build after the user confirms some options.


Jenkins configuration


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


 **Jenkins**


Jenkins > admin


 People

 Status

 Builds

 **Configure**

 My Views

 Credentials

Full Name

admin

Description

API Token

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.

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/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 a TESTPLAN variable, that in our case will be coming from Jira.

Jenkins > java-junit-calc >

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Project name

java-junit-calc

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

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

The final task submits the results linking the Test Execution to the Test Plan passed as an argument.

Post-build Actions

Xray: Results Import Task

X

JIRA Instance

xray-vm

⌵

Format

JUnit XML

⌵

Parameters

Execution Report File (file path with file name)

java-junit-calc/target/surefire-reports/TEST-com.xpand.java.CalcTe

Project Key

CALC

Test Execution Key

Test Plan Key

\$(TESTPLAN)

Test Environments

Revision

\$(BUILD_NUMBER)

Fix Version

v3.0

Power Actions configuration

We need to configure the Power Action custom field created earlier, namely the Condition, Screen, and Action Scripts; this from the custom field configuration as follows.

Condition Script

```
number ENABLED = 1;
number DISABLED = 2;
number HIDDEN = 3;

return ENABLED;
```

Screen Script

```
struct job {
    string name;
}

struct answer {
    job [] jobs;
}

persistent string defaultJobName = "";

string jenkinsBaseUrl = silEnv("JENKINS_BASE_URL");
string jenkinsUsername = silEnv("JENKINS_USERNAME");
string jenkinsPassword = silEnv("JENKINS_PASSWORD");

function getJenkinsJobs(){
    string requestURL = #{jenkinsBaseUrl} + "/api/json";

    HttpRequest request;
    HttpHeaders authHeader = httpBasicAuthHeader(jenkinsUsername, jenkinsPassword);
    request.headers += authHeader;
    HttpHeaders header = httpCreateHeader("Content-Type", "application/json");
    request.headers += header;

    //Post data and get response
    answer ans = httpPost(requestURL, request);
    logPrint("DEBUG", "answer: " + #{ans});
    return ans.jobs;
}

boolean isDisabled = false;
string [] jobs = getJenkinsJobs();
string [] ret = BA_setActionTitle("Trigger CI/CD build" + defaultJobName);
ret = BA_createSelectList("job", jobs, defaultJobName, isDisabled);
ret = arraysConcat(ret, BA_createSingleCheckbox("report to this test plan", true, isDisabled));
ret = arraysConcat(ret, BA_createSingleCheckbox("set as default job", false, isDisabled));
return ret;
```

Action Script

```
struct job {
    string name;
}

struct answer {
    job [] jobs;
}

string jenkinsBaseUrl = silEnv("JENKINS_BASE_URL");
string jenkinsUsername = silEnv("JENKINS_USERNAME");
string jenkinsPassword = silEnv("JENKINS_PASSWORD");
string token = silEnv("JENKINS_TOKEN");

//string jobName = "java-junit-calc";
string jobLabel = getElement(argv, 0);
string jobName = getElement(argv, 1);
string reportToTestPlanLabel = getElement(argv, 2);
boolean reportToTestPlan = BA_isChecked(argv, reportToTestPlanLabel); //getElement(argv, 3);
string setAsDefaultJobLabel = getElement(argv, 4);
boolean setAsDefaultJob = BA_isChecked(argv, setAsDefaultJobLabel); //getElement(argv, 5);
persistent string defaultJobName = "";

function triggerJenkinsBuild(string testPlanKey, string jobName){
    string requestURL = #{jenkinsBaseUrl} + "/job/" + #{jobName} + "/buildWithParameters?token=" + #{token} +
    "&TESTPLAN=" + #{testPlanKey};

    HttpRequest request;
    HttpHeaders authHeader = httpBasicAuthHeader(jenkinsUsername, jenkinsPassword);
    request.headers += authHeader;
    HttpHeaders header = httpCreateHeader("Content-Type", "application/json");
    request.headers += header;

    //Post data and get response
    string result = httpPost(requestURL, request);
    logPrint("DEBUG", "result: " + #{result});
    return true;
}

if (setAsDefaultJob){
    defaultJobName = jobName;
}
if (reportToTestPlan) {
    triggerJenkinsBuild(key, jobName);
} else {
    triggerJenkinsBuild("", jobName);
}
```

A new button will be available on the custom field. You'll need to add it to the view issue screen.



Bookstore / BOOK-30

Test Plan for v1.0

Edit

Comment

Assign

More ▾

Tests Pass

Tests failed

Adm

Details

Type: Test Plan

Status:

Priority: Trivial

Resolution:

Affects Version/s: None

Fix Version/s

Component/s: None

Labels: None

CI/CD: Trigger Jenkins

Trigger Bamboo

From the "job" dropdown you can select the job you want to run. You can also specify whether you want to report the results back to the current Test Plan and if you want to set that job as the default one for that Test Plan (so it pre-selected the next time you invoke it).

Trigger Bamboo plan and report back to Test Plan

In this case, we're going to give users the ability to trigger an existing job/project in Jenkins.

We'll obtain the list of available plans using Bamboo REST API which we'll present to the user. Then, we trigger the build after the user confirms some options.

Bamboo configuration

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

Build projects / Xray Automation Samples / java-junit-calc

←
✓
✓
✓
✓
✓
✓
✓
✓
✓
✓

Run
Actions

java-junit-calc

Plan Configuration

Stages & jobs1

Default Stage

default

Branches0

Plan details

Stages

Repositories

Triggers

Branches

Dependencies

Permissions

Notifications

Variables

Miscellaneous

Audit log

Variables

How to use variables

Variables substitute values in your task configuration and inline scripts. If the key contains the phrase 'password', like 'userpassword' the value will be masked with '*****'.

For task configuration fields, use the syntax \$(bamboo.myvariablename). For inline scripts, variables are exposed as shell environment variables which can be accessed using the syntax \$bamboo_MY_VARIABLE_NAME (Linux/Mac OS X) or %BAMBOO_MY_VARIABLE_NAME% (Windows).

Variable name

Value

Add

TESTPLAN

CALC-1200

The final task submits the results linking the Test Execution to the Test Plan passed as an argument.

Stages & jobs1

Default Stage

default

Branches0

Job details

Tasks

Requirements

Artifacts

Miscellaneous

Tasks

A task is a piece of work that is being executed as part of the build. The execution of a script, a shell command, an Ant Task or a Maven goal is about tasks.

You can use runtime, plan and global variables to parameterize your tasks.

Source Code Checkout

Checkout Default Repository

Maven 3.x

clean compile test

Final tasks

Are always executed even if a previous task fails

Xray: Results Import Task

Import JUnit XML

Add task

Xray: Results Import Task configuration

Task description

Import JUnit XML

Disable this task

JIRA instance*

Local JIRA

Format*

JUnit XML

Execution Report File (file path with file name)*

java-junit-calc/target/surefire-reports/

Project Key

CALC

Test Execution Key

Test Plan Key

\$(bamboo.TESTPLAN)

Power Actions configuration

We need to configure the Power Action custom field created earlier, namely the Condition, Screen, and Action Scripts; this from the custom field configuration as follows.

Condition Script

```

number ENABLED = 1;
number DISABLED = 2;
number HIDDEN = 3;

return ENABLED;

```

Screen Script

```
struct plan {
    string name;
    string key;
}

struct plans {
    plan [] plan;
}

struct answer {
    plans plans;
}

string bambooBaseUrl = silEnv("BAMBOO_BASE_URL");
string bambooUsername = silEnv("BAMBOO_USERNAME");
string bambooPassword = silEnv("BAMBOO_PASSWORD");
persistent string defaultBambooPlanName = "";

function getBambooPlans(){
    string requestURL = #{bambooBaseUrl} + "/rest/api/latest/plan.json";

    HttpRequest request;
    HttpHeaders authHeader = httpBasicAuthHeader(bambooUsername, bambooPassword);
    request.headers += authHeader;
    HttpHeaders header = httpCreateHeader("Content-Type", "application/json");
    request.headers += header;

    //Post data and get response
    answer ans = httpGet(requestURL, request);
    logPrint("DEBUG", "answer: " + #{ans});

    string [] jobs;
    for(plan plan in ans.plans.plan) {
        logPrint("DEBUG", "plan_key: " + plan.key);
        jobs = addElement(jobs, plan.key);
    }

    return jobs;
}

boolean isDisabled = false;
string [] jobs = getBambooPlans();
string [] ret = BA_setActionTitle("Trigger CI/CD build"+defaultBambooPlanName);
ret = BA_createSelectList("plan", jobs, defaultBambooPlanName, isDisabled);
ret = arraysConcat(ret, BA_createSingleCheckbox("report to this test plan", true, isDisabled));
ret = arraysConcat(ret, BA_createSingleCheckbox("set as default plan", false, isDisabled));
return ret;
```

Action Script

```
string bambooBaseUrl = silEnv("BAMBOO_BASE_URL");
string bambooUsername = silEnv("BAMBOO_USERNAME");
string bambooPassword = silEnv("BAMBOO_PASSWORD");

string planLabel = getElement(argv, 0);
string planName = getElement(argv, 1);
string reportToTestPlanLabel = getElement(argv, 2);
boolean reportToTestPlan = BA_isChecked(argv, reportToTestPlanLabel); //getElement(argv, 3);
string setPlanAsDefaultLabel = getElement(argv, 4);
boolean setPlanAsDefault = BA_isChecked(argv, setPlanAsDefaultLabel); //getElement(argv, 5);
persistent string defaultBambooPlanName = "";

function triggerBambooBuild(string testPlanKey, string planName) {
    string requestURL = #{bambooBaseUrl} + "/rest/api/latest/queue/" + #{planName};
    // "default&ExecuteAllStages=true&bamboo.TESTPLAN=#{testPlanKey}"

    HttpRequest request;
    HttpHeaders authHeader = httpBasicAuthHeader(bambooUsername, bambooPassword);
    request.headers += authHeader;
    HttpHeaders header = httpCreateHeader("Content-Type", "application/x-www-form-urlencoded");
    request.headers += header;

    request.parameters += httpCreateParameter("default", "true");
    request.parameters += httpCreateParameter("ExecuteAllStages", "true");
    request.parameters += httpCreateParameter("bamboo.TESTPLAN", testPlanKey);

    //Post data and get response
    string result = httpPost(requestURL, request);
    logPrint("DEBUG", "result: " + #{result});
    return true;
}

if (setPlanAsDefault) {
    defaultBambooPlanName = planName;
}

if (reportToTestPlan) {
    triggerBambooBuild(key, planName);
} else {
    triggerBambooBuild("", planName);
}
```

A new button will be available on the custom field. You'll need to add it to the view issue screen.



Bookstore / BOOK-30

Test Plan for v1.0

Edit

Comment

Assign

More

Tests Pass

Tests failed

Admin

Details

Type: Test Plan

Status: **TESTING** (View Workflow)

Priority: ☐ Trivial

Resolution: Unresolved

Affects Version/s: None

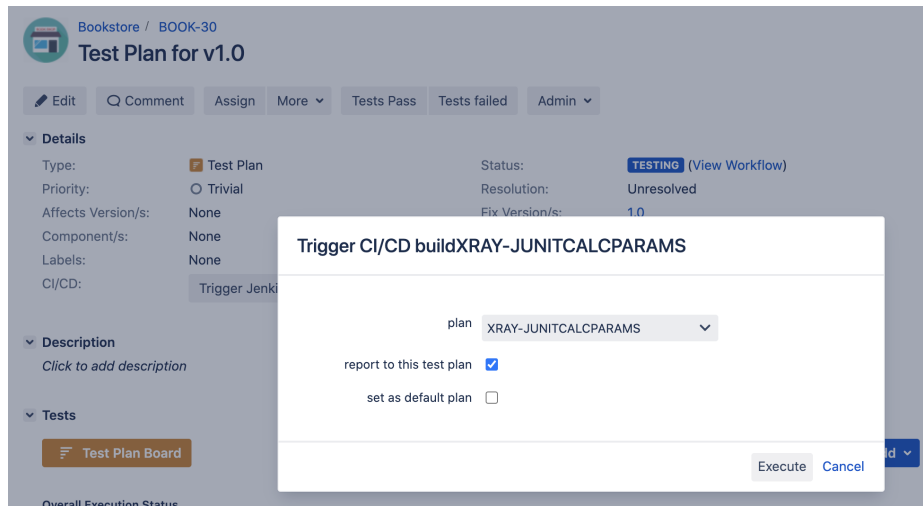
Fix Version/s: 1.0

Component/s: None

Labels: None

CI/CD:

From the "plan" dropdown you can select the plan you want to run. You can also specify whether you want to report the results back to the current Test Plan and if you want to set that job as the default one for that Test Plan (so it pre-selected the next time you invoke it).

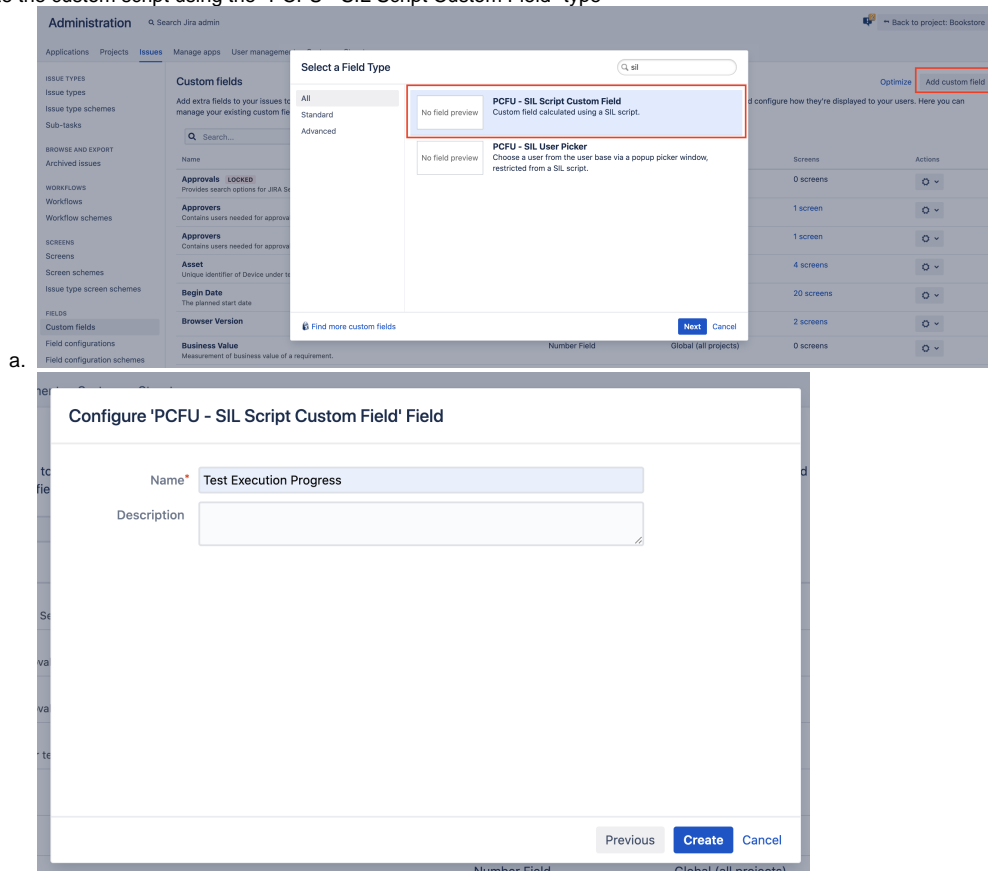


Display progress information, as text, on Test Execution issues

The idea is to display information about the number of passed, failed, etc tests as text, on Test Execution issues using a specific custom field.

Therefore, we'll use the Power Custom Fields app.

1. create the custom script using the "PCFU - SIL Script Custom Field" type



2. configure the custom field by defining the respective SIL script; in this case we'll just return the value of "Test Execution Status" custom field, as it already provides a textual representation with what we need

Administration Search Jira admin

Applications Projects **Issues** Manage apps User management System Structure

ISSUE TYPES
Issue types
Issue type schemes
Sub-tasks

BROWSE AND EXPORT
Archived issues

WORKFLOWS
Workflows
Workflow schemes

SCREENS
Screens
Screen schemes
Issue type screen schemes

FIELDS
Custom fields

Configure Custom Field: Test Execution Progress

Below are the Custom Field Configuration schemes for this custom field. Schemes are applicable for various issues types in a project level schemes will over-ride global ones.

- [Add new context](#)
- [View Custom Fields](#)

Default Configuration Scheme for Test Execution Progress

Default configuration scheme generated by Jira

Applicable contexts for scheme: [Edit Configuration](#)

Issue type(s):
Global (all issues)

SIL Script:

```
Thread Local Caching: false
Format Date: none
return #{Test Execution Status};
Edit SIL Script
```

a.

Now you can add this custom field on the Test Execution view issue screen if you want. In this specific case, it would be a bit redundant though, as the progress bar below shows those values.

Bookstore / BOOK-31

Test Execution for Test Plan BOOK-30

[Edit](#) [Comment](#) [Assign](#) [More](#) [Close Issue](#) [Reopen Issue](#) [Admin](#)

Details

Type:	Test Execution	Status:	RESOLVED (View Workflow)
Priority:	Major	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	1.0
Component/s:	None		
Labels:	default		
Test Plan:	BOOK-30		
Test Environments:	None		
Revision:	65d64uyg7t6r		
Test Execution Progress:	PASS : 15(93.75%), FAIL : 1(6.25%), ABORTED : 0(0.0%), PENDING : 0(0.0%), EXECUTING : 0(0.0%), BLOCKED : 0(0.0%), TODO : 0(0.0%)		

Description
[Click to add description](#)

Tests [+ Add](#)

Overall Execution Status

15 PASS 1 FAIL

You can also add it as a column on panels that show Test Executions (e.g. on the Test Plan).

