



JIRA TIPS & SCRIPTS



TABLE OF CONTENTS

Jira Quick Tips.....	4
Jira Useful Shortcuts.....	6
Selection of Jira Rules and Other Automation App Scripts	9
Tutorials on Tempo Automation Rules.....	10
A Collection of ScriptRunner & Power Scripts.....	18
About Tempo	25



INTRODUCTION

Whether you're starting out in Jira or have been using it for years, there's always plenty more to learn. It's a flexible software that you can customize to your heart's content. A wide range of specialized add-ons and scripts support the advanced functionality of the tool.

With all the available options, it can be tough to know where to start. This document lays out some handy tips and recommendations for first-timers and veteran Jira users alike.

If you're looking for insight into how to improve your Jira setup and experience, read on!

JIRA QUICK TIPS

Efficiency is the name of the game with these helpful tips designed to save you time and headache.


Tips	Description
Use shortcuts	Keyboard shortcuts save time and effort. See our Jira shortcuts selection in the next chapter .
Set up filter subscriptions as reminders	Get your to-do list under control by creating a filter subscription. Here's how: First, create a Jira issue and assign it to yourself. Set a due date, then create a filter to find all issues due in the next X days. Save the filter and create a filter subscription to send an email when a due date is approaching.
Create a Browser Search Engine	In Chrome, add a custom search engine by clicking > Settings > Manage search engine, then add a new search engine.
Use time tracking tools for Jira	With sophisticated time tracking and reporting apps like Tempo Timesheets, you can keep track of billable hours, capitalized and operational expenses, and get all kinds of helpful insights into your company's operations.
Use in-line issue create for the fastest way to create new issues	You can create new Jira issues in-line on the backlog and the board (available in next-gen projects). When you are creating issues in-line, those new issues inherit all selected filters (labels, versions, assignees, etc.) and epic links, saving you loads of time!
Add issue keys to your branch and commit messages	If you are using Git as your version control system, make sure to add your issue key to branch name and commit messages. A good convention for branch naming is <prefix>/<issue-key>_<description>. If your team is using Bitbucket, you can create your branch from the issue screen which automatically creates a branch name using this convention.
If you work on multiple projects, you can create your own board to keep track of all your tickets	(This function does not work with next-gen projects).

Tips	Description
Save your searches to use later	If you are creating custom searches via JQL or Advanced searches (more on JQL here), make sure to save them. Saved searches are available throughout Jira to filter views (like on the board). You can even create custom boards from saved searches!
Add search results to Dashboards	<p>Created a powerful search? Add it to a dashboard to bring it to life! For example, keep track of high-priority tickets by creating a filter with JQL.</p> <ul style="list-style-type: none"> ◆ Add the Filter Results gadget to your dashboard ◆ Save the JQL above as a filter and enter its name in the Saved Filter field
Use automation apps	You can really push JSW and JSD to the next level with Automation for Jira, Scriptrunner, Power Scripts, and other automation apps. For example, you can create recurring activities on a schedule, or use Epics to create common tasks to maintain process adherence, meaning when you create an Epic meeting certain criteria then all associated tasks and assignments are created.
Use Jira Query Language (JQL)	JQL is a powerful way to search for your issues in Jira. JQL narrows your search results and ensures you have the relevant fields on display. You can then bring them into your dashboards to report lots of different information in different ways. For example, you can do a 2-dimensional chart of issues completed by application (custom field), and also show it graphically with a pie chart. Management teams find a lot of value in this function and are often adding/updating the dashboards to keep the team performance, to show current and completed work, and ensure nothing slips through the cracks.


There are many more tips out there for Jira, but start with these and they will certainly serve you well.


JIRA USEFUL SHORTCUTS

Everyone loves keyboard shortcuts. They just make life so easy and efficient. Thankfully, Jira has plenty of them, and so do [Tempo Timesheets](#) and [Tempo Planner](#).

	Shortcuts	Action
Jira Global	gd *	Go to dashboard
	Gp*	Browse to a project
	ga* or gh*	Go to agile
	gi *	Find issues
	gg *	Administration quick search
	/ *	Quick search
	c *	Create an issue
	.	Open the operations dialog. The fastest way to navigate Jira, give it a try. <i>(available in classic projects)</i>
	?	Open shortcut help (on Tempo for Server/Data Center)
	CTRL+ALT+S	Form submit

* on Tempo for Server/Data Center

	Shortcuts	Action
Jira Navigating Issues	o Enter	View selected issue
	j	Next issue
	k	Previous issue
	[Dock/undock the filters panel
	n	Next activity
	p	Previous activity
	f	Focus search field
	u	Search for issues
	t	Switch filter view
	Esc	Cancel the changes to content of a field being edited in the Detail View. The 'Esc' key can also be used to Close or Cancel dialog boxes.
	z	Detail view order by
Jira issue actions	e	Edit issue (on Server/Data Center)
	a	Assign issue to someone
	m	Comment on issue
	l	Edit issue labels
	,	Just to fields for editing (on Server/Data Center)
	i	Assign to me

	Shortcuts	Action
Tempo Timesheets shortcuts	g then p	Browse to the current project
	Cmd (Mac) or Ctrl (Windows) + drag	Copy a time record from one date to another in the My Work Calendar
	g then a, g then h	Go to Agile board
	g then t (on Server and Data Center)	Go to My Work from anywhere in Jira or Tempo
	g then i	Go to the issue navigator
	Tab	Move among fields in the Log Time form
	g then g	Go to the administration search dialog. This is available only if you have the Jira administrator permission.
	w	Open the Log Time form from anywhere in Jira or Tempo. Obs.: On cloud, in the Jira issue view, the “w” shortcut turns on/off the “watch” option in the issue.
	Cmd+Enter (Mac) or Ctrl+Enter (Windows)	Submit the time record from the Log Time form
Tempo Planner shortcuts	Shift + drag	Copy the selected plan to another date or resource in the Team Planning Timeline view
	Delete	Delete the selected plan in the Team Planning Timeline view. Press Enter to confirm or Esc to cancel.
	Esc	Escape the selected plan’s details in the Team Planning Timeline (on Server/Data Center it also works in the Resource Planning view)
	Cmd (Mac) or Ctrl (Windows) + click	Split the selected plan where you click it in the Team Planning timeline

Make use of these shortcuts and they will quickly become second nature.

SELECTION OF JIRA RULES AND OTHER AUTOMATION APP SCRIPTS

We don't know anyone using Jira these days without automation - it's an essential part of extending Jira. One of the apps we've seen growing by leaps and bounds is, of course, Automation for Jira!

About the Automation for Jira App

Back in 2019, Atlassian acquired [Automation for Jira](#), an app that had gained awards such as [Fastest Server Growth](#).

And then they made it free on Jira Cloud!

Premium users have 1000 global and multi-project rules per paid user per month. (i.e 200 users in Jira Cloud Premium will have 200,000 monthly global/multi-project rules per month). This is pooled across all Jira tools and all users. Customers on free and standard plans have access to 100 and 500 global and multi-project rules per month respectively. [Source](#) (still available on Server as a [paid app](#))

TIP



Estimate how many times your rule will run by looking at your trigger. If it's an event like "Issue Created", go back and search to see the average number of issues created over the last few months. These rules add up quickly and can max out if you don't plan ahead!

87% of automation users say it helps them scale the way their organization works
[Source](#)

Automation Playground & Templates

The automation playground is a sandbox environment with hundreds of automation templates. Recreate rules in your own Jira instance by using the templates.

Try out some rules yourself by checking out the ["Automation Playground"](#). You can also watch the Atlassian University [webinars](#) to learn more.

Automation						Create rule
Filter rules	Name	Labels	Owner	Project	Enabled	
All rules	Auto-assign high priority Jira issues	MP AA	Steve Ruler	Multiple projects	✓	
Global rules	Auto-assign Jira issues upon transition	MP AA JS	5da0ddc650... (deleted user?)	Global	✓	
Most popular	Auto-close old Jira Service Desk support issues	MP AA JS	Steve Ruler	Steve Loves Scrum	✓	
Distributed teams	Auto close parent issue when all sub-tasks are done (sync)	MP JS	Steve Ruler	Global	✓	
Notifications	Automatically add 3 sub-tasks on issue creation	MP	Steve Ruler	Multiple projects	✓	
DevOps	Close duplicate Jira issues	MP	Steve Ruler	Global	✓	
Jira Business	Close parent when all sub-tasks have been completed	MP JS A	Steve Ruler	Global	✓	
Jira sync	Close stories when epic is marked as done	MP JS A	Steve Ruler	Global	✓	
Agile	Close sub-tasks when parent is completed	MP JS	Steve Ruler	Global	✓	
Auto assign	If a comment is added to an issue and the reporter is in admin, then transition the issue 'DONE'	MP	Steve Ruler	Global	✓	
Jira Service Desk	Link mentioned issue	MP DT LI	Steve Ruler	Global	✓	
Software Teams	Notify via email for high priority issues	MP DT N	Steve Ruler	Steve's Software House	✓	
Jira business	Re-open issue on customer comment	MP DT	Steve Ruler	Steve's Service Desk	✓	

Tutorials on Tempo Automation Rules

At [Tempo](#), we've created a few automation rules based off of conversations we've had with our customers to make life easier and let the machine do the work!

Feel free to copy and tweak the following rules and share your creations by tagging us on [Twitter](#). Note that to be able to execute these automation rules, you need to have the Tempo app installed in your Jira instance.

Start a 30-day trial of Tempo here:

- ◆ [Try Tempo Timesheets](#)
- ◆ [Try Tempo Planner](#)

Automation Rules for Tempo Timesheets

Set a Tempo team on Jira issue assignment.

There might be cases where we want to [assign a Jira issue to a specific Tempo Team](#) based on certain conditions. In the example below we use the Jira issue assignment as trigger to run of an automation rule to set a Tempo Team on the Jira issue. Further we can implement a what-if scenario by setting the Team name based on the assignee user or the assignee belonging to a Jira group.

Since the Tempo Team field can't be set out of the box we will need to leverage the advances options when editing the Jira issue in an automation. Therefore we need the Tempo team customfield ID. Jira custom fields can be retrieved from the Jira API

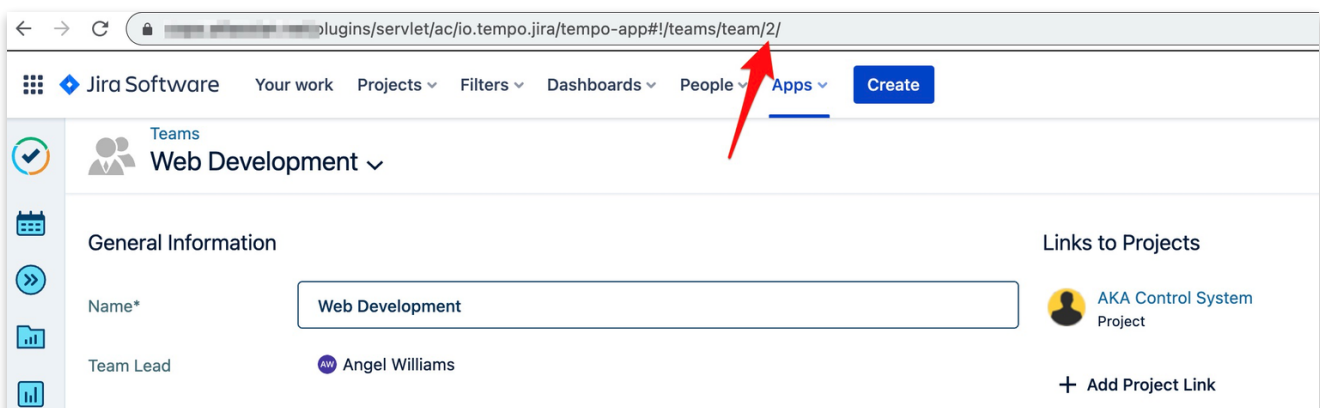
```
{YourJiraCloudUrl}/rest/api/3/field
```

In the response search for “Team”. This should give you something like:

```
{
  "id": "customfield_11718",
  "key": "io.tempو.jira__team",
  "name": "Team",
  "untranslatedName": "Team",
  "custom": true,
  "orderable": true,
  "navigable": true,
  "searchable": true,
  "clauseNames": [
    "cf[11718]", "Team"
  ],
  "schema": {
    "type": "option2",
    "custom": "com.atlassian.plugins.atlassian-connect-plugin:io.
tempو.jira__team",
    "customId": 11718
  }
},
```

That means in our case the Tempo Team customfield has the ID “11718”.

Before we go into Automation for Jira we will also need the respective Tempo Team IDs as we can't use the Tempo Team names in Automation for Jira. We can get the Team ID by navigating to the Tempo Team and look for the ID in the url.



Or we can leverage the Tempo Team REST API to get the Tempo Teams, their names and IDs.

```
https://api.tempio.io/core/3/teams
```

This will get us a response as:

```
{
  "self": "https://api.tempio.io/core/3/teams/2",
  "id": 2,
  "name": "Web Development",
  "summary": "",
  "lead": {
    "self": "https://cops.atlassian.net/rest/api/2/user?accountId=5cb085549b0de43f0aea7154",
    "accountId": "5cb085549b0de43f0aea7154",
    "displayName": "Angel Williams"
  },
  "program": null,
  "links": {
    "self": "https://api.tempio.io/core/3/teams/2/links"
  },
  "members": {
    "self": "https://api.tempio.io/core/3/teams/2/members"
  },
  "permissions": {
    "self": "https://api.tempio.io/core/3/teams/2/permissions"
  }
},...
```

Make sure that the Tempo Team is linked to the Jira project where you want to use the Jira automation rule for. The Tempo Team can only be set on the Jira issue when the Tempo Team is linked to the Jira project.

Once we ensured that we have everything we need to ensure all settings are correct we can go ahead with Automation for Jira.

Our rule looks like this:

The screenshot shows the Jira Automation interface for a rule titled "Set the Tempo Team field on assignment". The rule is in a "DRAFT" state. The left sidebar shows the rule flow: "When: Issue assigned" (Rule is run when an issue is assigned to a user.), "If: matches" (User: Amy Mitchell), "Then: Edit issue fields" (Advanced), "Add component", "Else-if: matches" (jira-users), "Then: Edit issue fields" (Advanced), and "Add component". The main area shows the configuration for the "If block". The "User condition" is set to "User" with the value "Assignee". The "Check to perform" is set to "is". The "Criteria" are set to "Amy Mitchell". There is a "+ Add additional criteria" link. At the bottom right of the configuration area are "Cancel" and "Save" buttons.

The trigger for the rule is when a user is assigned to a Jira issue or the assignee changes.

We implemented in the example a condition rule checking on the assignee. Means we want to set a different Tempo Team based on the assignee of the Jira issue. Without going into too much details you can check the value against a specific user or if the assignee belongs to a certain Jira group. Just have in mind that the if-block is exited once a condition is met.

After the condition check we finally assign the Tempo Team to the issue by setting the Tempo Team custom field.

The screenshot shows the Jira Automation interface for the same rule. The left sidebar shows the rule flow: "When: Issue assigned" (Rule is run when an issue is assigned to a user.), "If: matches" (User: Amy Mitchell), and "Then: Edit issue fields" (Advanced). The main area shows the configuration for the "Edit issue" block. The "Choose fields to set..." dropdown is visible. Under "More options", "Send notifications?" is checked, and "This rule should send emails. Rule actor must be an admin or project admin." is also checked. The "Additional fields" section contains a JSON snippet:

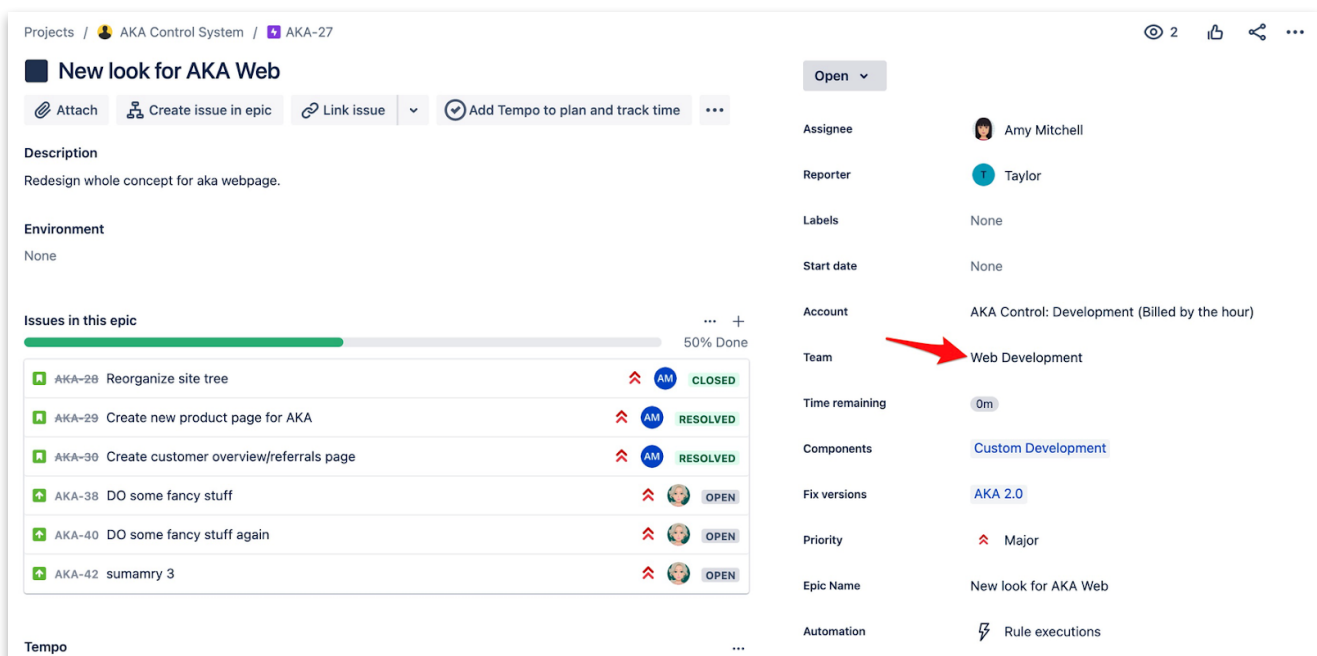
```
{
  "fields": {
    "customfield_11718": "2"
  }
}
```

 At the top right of the configuration area is a "Publish rule" button.

In the “More Options” section you set the values based on the Jira customfield ID and the Tempo Team ID that we have retrieved above. Make sure that you escape the Tempo Team ID.

```
{
  "fields": {
    "customfield_11718": "2"
  }
}
```

That’s all. As always check the audit logs if any problems occur. The automation should now set the Tempo Team customfield.



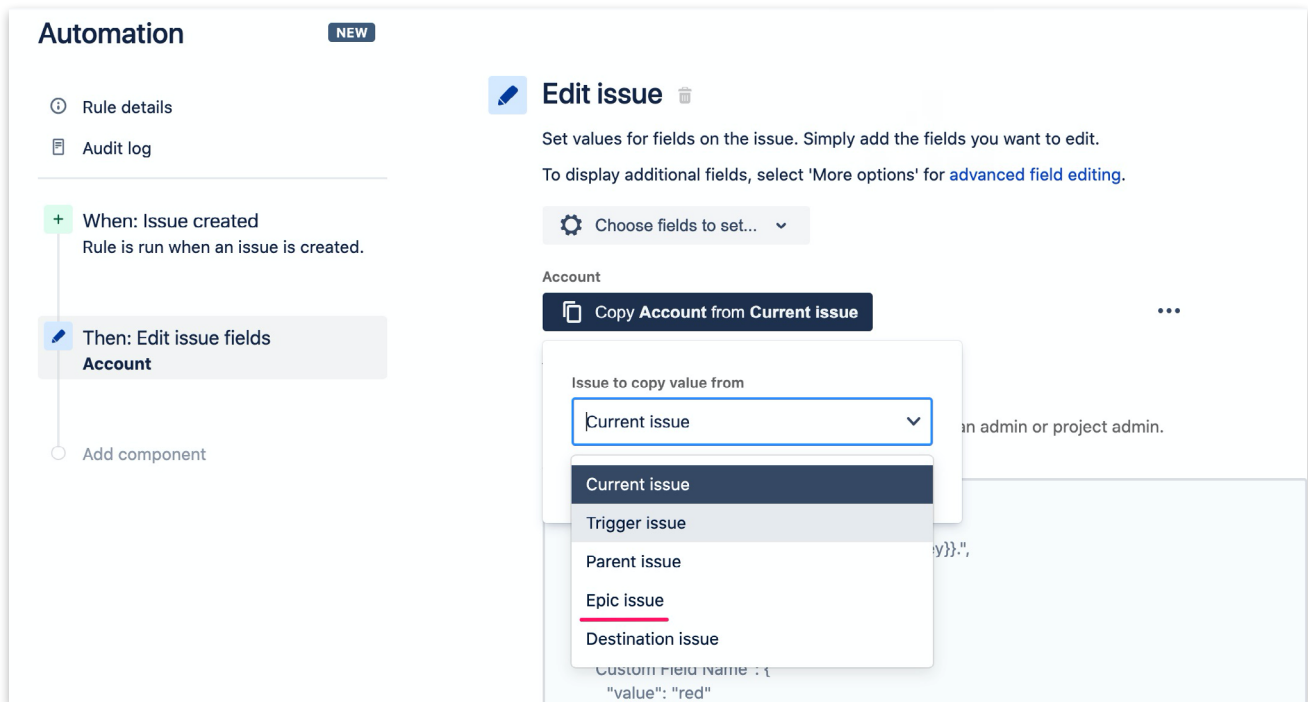
Set an automated Tempo Account.

There might be a need to [set a Tempo Account](#) on an issue based on certain rules. With Automation for Jira this can easily be achieved and there are several options to choose from.

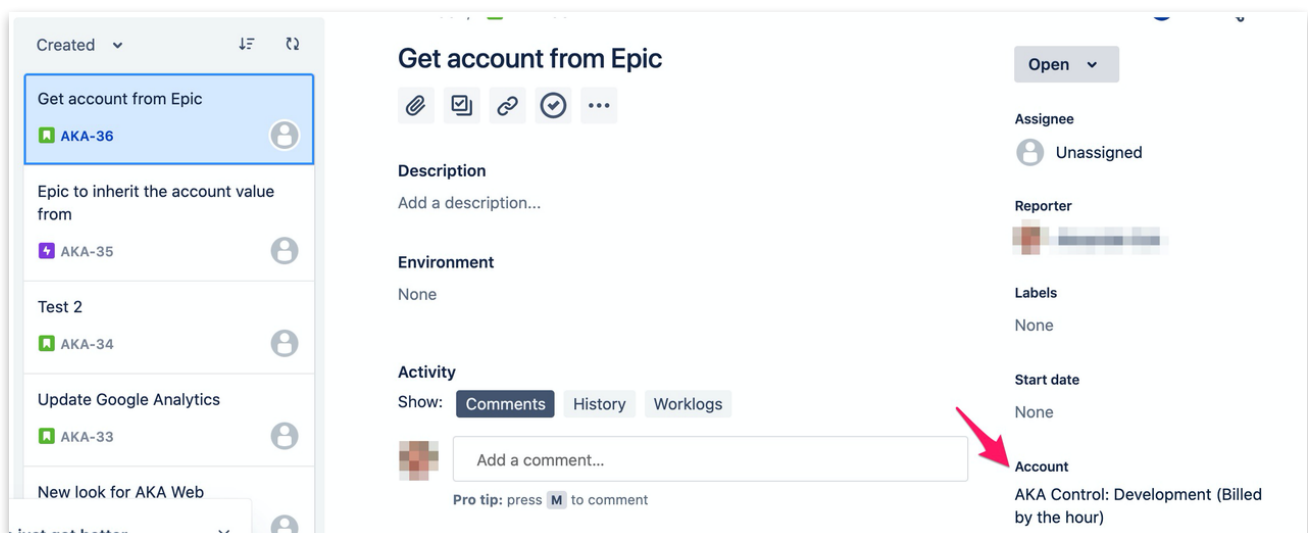
In this tutorial we focus on setting a Tempo Account in the event of the creation of a Jira issue. We will set the Tempo Account based on the Account that is set on the Epic for the issue we will create.

Instructions

1. Set up an automation rule under projects > settings > automation.
2. As the trigger for the automation choose “Issue created”.
3. Add a new action.
4. Select “Edit Issue” and in the field, choose the Tempo “Account” field.
5. In Operations pick “Copy from” and in the dropdown where the value to copy from choose “Epic issue”.



6. Save and publish the automation.
7. Create a new issue in the project and select an Epic link where the Tempo Account value has been set.
8. You will notice that the Tempo Account value has been inherited from the Epic.



Log automated work on a Jira transition

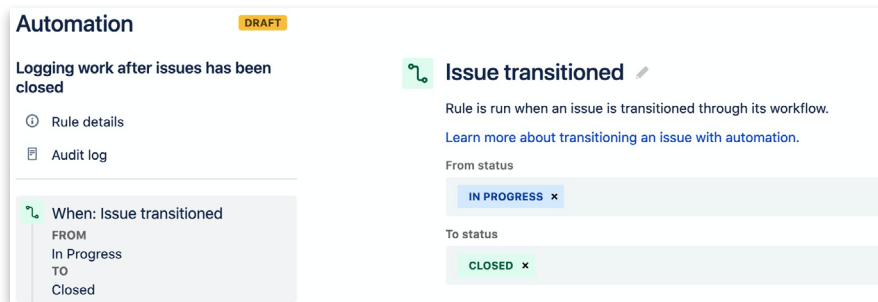
There might be a need to automate logging work triggered by an action in Jira. E.g. I need to make sure that there has been logged work on Jira transition. This can easily be achieved with Automation for Jira Cloud.

In our [example](#), we will leverage the Tempo REST APIs ([creating a worklog](#)) in order to do so. In order to consume the Tempo REST API you will need an [Access token](#).

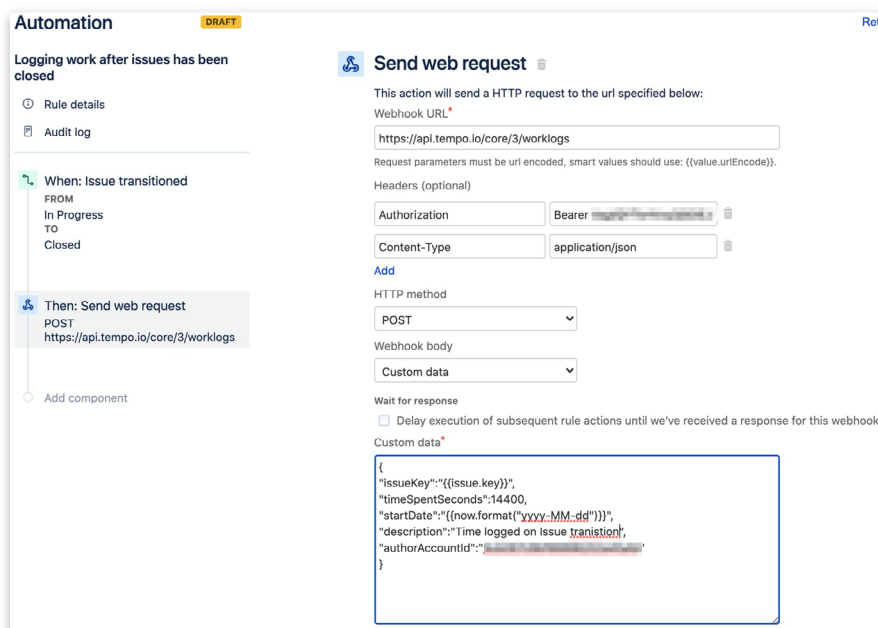
Instructions

Follow the steps

1. From your Jira project settings navigate to Automations.
2. Define the trigger that will fire the automation. There are many, many options to do so. In our example we choose a workflow transition.



3. Next we define the action that the automation rule should perform for us. From the list of actions available pick "Send web request".
4. Now fill in the required fields similar to the screenshot below. Jira automation offers many (endless) [smart values](#) to be used in the payload of our request.



5. Before you publish the automation you can validate the action to make sure it works properly. You will see a detailed response in case of a failed validation.

Automation Rules for Tempo Planner

Create a Tempo plan on user assignment

When you use Tempo Planner and want an automated process to [create a Tempo plan](#) once a user is assigned to a Jira ticket you can use Automation for Jira in order to achieve this.

Make sure you have an [access token](#) with the “Plan time” scope before you proceed with the steps below.

Instructions

From Jira project settings go to Automations and create a new automation for your Jira project/s.

1. As the trigger for your automation select “Issue assigned”
2. Next we select an action. From the actions available pick “Send web request”. Fill in the fields like shown in the screenshot below.
3. Check your automation with the validation option on the bottom.

The screenshot shows the Jira Automation configuration interface. The automation is titled "Create a Tempo Plan after user is assigned" and is currently "ENABLED". The trigger is "When: Issue assigned". The action is "Send web request" with the following configuration:

- Webhook URL: `https://api.tempo.io/core/3/plans`
- Request parameters: (empty)
- Headers: Authorization (redacted), Content-Type: `application/json`
- HTTP method: `POST`
- Webhook body: Custom data
- Wait for response: Delay execution of subsequent rule actions until we've received a response for this webhook
- Custom data:

```
{
  "startDate": "{{(now.jiraDate}}",
  "endDate": "{{(issue.dueDate.jiraDate}}",
  "description": "Plan made by automation tool after setting an assignee",
  "plannedPerDaySeconds": {{(=#)}}{{(issue.fields.timeestimate)} /
  {{now.diff(issue.dueDate).days}}{)},
  "accountId": "{{(issue.assignee.accountid}}",
  "issueKey": "{{(issue.key}}"}
}
```

For the automation example above to work you will need to have a due date and a remaining estimate set on your Jira issue.

When you have done everything correct and your automation was executed you should see the plan created by the Jira automation rule.

The screenshot shows the Tempo Resource Planning interface. The view is for the period "30/Aug/20 - 3/Oct/20". A resource named "Member" is shown with a total allocation of "166h 40m". The resource is assigned to a plan titled "Do an automated plan" (AKA-37) with a status of "OPEN". The plan is scheduled for execution on August 30th, 31st, and September 1st, with a duration of 1.3 hours per day. The plan description is "Plan made by automation tool after setting an assignee".

A Collection of ScriptRunner & Power Scripts

We've been on the automation bandwagon for a while now, collaborating with [Adaptavist](#) for a ScriptRunner Tempo library of scripts, and a PowerScripts library as well.

We selected our 3 favourite scripts for Server (see below). however feel free to check out the [complete script library](#) to help you automate your work.

Filtering by a worklog attribute

The script will search/return all worklogs on an issue with a certain worklog attribute.

```
import com.atlassian.jira.component.ComponentAccessor
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.tempoplugin.core.datetime.api.TempoDate
import com.tempoplugin.worklog.v4.rest.InputWorklogsFactory
import com.tempoplugin.worklog.v4.rest.TimesheetWorklogBean
import com.tempoplugin.worklog.v4.services.WorklogService
import org.apache.log4j.Level
import org.apache.log4j.Logger

import com.tempoplugin.core.workattribute.api.WorkAttributeService
import com.tempoplugin.core.workattribute.api.WorkAttributeValueService
import com.atlassian.jira.issue.worklog.Worklog
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.MutableIssue

def myLog = Logger.getLogger("com.onresolve.jira.groovy")

myLog.setLevel(Level.DEBUG)

@WithPlugin("is.origo.jira.tempo-plugin")
@PluginModule
WorkAttributeService workAttributeService

@PluginModule
WorkAttributeValueService workAttributeValueService

def worklogManager = ComponentAccessor.worklogManager
def IssueManager = ComponentAccessor.getIssueManager()
```

```

MutableIssue issue = IssueManager.getIssueObject("WIKK-20")
def worklogs = worklogManager.getByIssue(issue)
// Filter the worklogs which are marked as "Remote"
def remoteLogs = worklogs.findAll { worklog ->
def attribute = workAttributeService.getWorkAttributeByKey("_WorklogCategory_").returnedValue
def attributeValue = workAttributeValueService.getWorkAttributeValueByWorklogAndWorkAttribute(worklog.
id, attribute.id).returnedValue
if (attributeValue) {
    //myLog.info("Worklog attribute value: " + attributeValue.value)
    if(attributeValue.value=="Development") {
        //myLog.info("Worklog attribute value: " + attributeValue.value)
        attributeValue
    }
}
}
// Sum the remote time in seconds. If there aren't any remote worklogs, just return null
remoteLogs.sum { Worklog worklog ->worklog.timeSpent} as Long

```

Create a TempoAccount Using the REST API

Create a Tempo account (a way to track time across multiple teams and projects) using the Tempo REST API. Utilising the Tempo REST Endpoint, this script uses basic authentication in the headers of the request to authenticate the user. The authenticated user must have permission to create a Tempo account. Read more about permissions in Tempo in this article.

Note: *this is a Server script*

```

import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.config.properties.APKeys
import groovyx.net.http.ContentType
import groovyx.net.http.EncoderRegistry
import groovyx.net.http.HttpResponseDecorator
import groovyx.net.http.RESTClient
// the user-defined property where the user name and password are stored into
final String userPropertyKey = "jira.meta.basicAuthCreds"

```

```

def loggedInUser = ComponentAccessor.jiraAuthenticationContext.loggedInUser
def credentials = ComponentAccessor.userPropertyManager.getPropertySet(loggedInUser).
getString(userPropertyKey)
def baseUrl = ComponentAccessor.applicationProperties.getString(APKeys.JIRA_BASEURL)

def data = [
    name : "Account",
    key : "12345",
    lead : [
        key : "admin",
        username: "admin"
    ],
    status: "OPEN"
]

def client = new RestClient(baseUrl)
client.encoderRegistry = new EncoderRegistry(charset: 'UTF-8')
client.setHeaders([
    Authorization : "Basic ${credentials.bytes.encodeBase64().toString()}",
    "X-Atlassian-Token": "no-check"
])
client.handler.success = { response, json ->
    log.debug " Account successfully created"
    json['id']
}
client.handler.failure = { HttpResponseDecorator response ->
    log.error response.entity.content.text
}
client.post(
    path: '/rest/tempo-accounts/1/account/',
    contentType: ContentType.JSON,
    body: data
)

```

Add Tempo Planning Information

Gather all existing Tempo plans for a Jira issue (defined by its ID from Jira, and interpreted as the planItemId in Tempo) for a given period (start date to end date) and get total time planned, using the Tempo REST API. This script uses the Tempo REST Endpoint with basic authentication in the headers of the request. The authenticated user must have permission to view the plans for users; otherwise, the API does not return plans.

Note: *this is a Server script*

```
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.config.properties.APKeys
import com.atlassian.jira.event.type.EventDispatchOption
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.tempoplugin.planner.api.event.AllocationEvent
import groovyx.net.http.ContentType
import groovyx.net.http.HttpResponseDecorator
import groovyx.net.http.RESTClient
import groovyx.net.http.EncoderRegistry
import java.time.Duration
import java.time.LocalDate
import java.time.format.DateTimeFormatter
import java.util.concurrent.TimeUnit

@WithPlugin('com.tempoplugin.tempo-plan-core')
// The user-defined property where the user name and password are stored into
final userPropertyKey = 'jira.meta.basicAuthCreds'
final plannedTimeField = 'Planned Time'
/**
 * Helper method to print Duration time as hours, minutes and seconds.
 * @param duration Time to be formatted.
 * @return Duration formatted as String.
 */
String prettyPrintDuration(Duration duration) {
    String.format("%s h ${duration.toMinutes() == 0 ? '%s m' : ''} ${duration.seconds == 0 ? '%s s' : ''}",
duration.toHours(),
        duration.toMinutes() - TimeUnit.HOURS.toMinutes(duration.toHours()),
        duration.seconds - TimeUnit.MINUTES.toSeconds(duration.toMinutes())).trim()
}
}
```

```

def loggedInUser = ComponentAccessor.jiraAuthenticationContext.loggedInUser

def credentials = ComponentAccessor.userPropertyManager.getPropertySet(loggedInUser).
getString(userPropertyKey)

def baseUrl = ComponentAccessor.applicationProperties.getString(APKeys.JIRA_BASEURL)

def client = new RestClient(baseUrl)

client.encoderRegistry = new EncoderRegistry( charset: 'UTF-8' )

client.setHeaders([
    Authorization    : "Basic ${credentials.bytes.encodeBase64().toString()}",
    "X-Atlassian-Token": "no-check"
])

client.handler.failure = { HttpResponseDecorator response ->
    log.error response.entity.content.text
    []
}

client.handler.success = { resp, reader ->
    [response: resp, reader: reader]
}

def today = LocalDate.now()
def todayPlus90Days = today.plusDays(90)

def event = event as AllocationEvent

def allocationResult = (client.get(
    path: '/rest/tempo-planning/1/allocation',
    query: [
        planItemId : event.allocation.planItemId,
        planItemType: 'ISSUE',
        assigneeType: 'user',
        startDate  : DateTimeFormatter.ISO_LOCAL_DATE.format(today),
        endDate    : DateTimeFormatter.ISO_LOCAL_DATE.format(today)
    ]
) as Map).reader as List<Map>

if (!allocationResult) {
    log.error "There is no allocation result related with the plan"
    return
}

```

```

def taskKey = (allocationResult.first()?.planItem as Map)?.key
if (!taskKey) {
    log.error "There is no issue related with the plan"
    return
}
def planSearchResult = (client.post(
    path: '/rest/tempo-planning/1/plan/search',
    contentType: ContentType.JSON,
    body: [
        from : DateTimeFormatter.ISO_LOCAL_DATE.format(today),
        to   : DateTimeFormatter.ISO_LOCAL_DATE.format(todayPlus90Days),
        taskKey: [taskKey]
    ]
) as Map).reader as List<Map>

if (!planSearchResult) {
    log.error "There is no time planned related with the plan"
    return
}
def totalSeconds = planSearchResult.sum { (it as Map).timePlannedSeconds }
def duration = Duration.ofSeconds(totalSeconds as Long)
def plannedTime = prettyPrintDuration(duration)
def cfManager = ComponentAccessor.customFieldManager
def plannedTimeCf = cfManager.getCustomFieldObjectsByName(plannedTimeField).first()
def issueManager = ComponentAccessor.issueManager
def issue = issueManager.getIssueObject(taskKey as String)
issue.setCustomFieldValue(plannedTimeCf, plannedTime)
issueManager.updateIssue(loggedInUser, issue, EventDispatchOption.DO_NOT_DISPATCH, false)

```

Complete Script Library

1. [Creating a worklog with worklog attributes using the REST API](#)
2. [Dynamic work attribute based on the user selection](#)
3. [Event listeners](#)
4. [Filtering by a worklog attribute](#)
5. [Searching for worklogs](#)
6. [Summing worklogs of an Account](#)
7. [Tempo Account information](#)
8. [Tempo Team custom field](#)
9. [Create a Tempo Account Using the REST API](#)
10. [Display Account Lead on Jira Issue](#)
11. [Display Formatted Account Metadata on Jira Issues](#)
12. [Get Tempo Plans Using the REST API for a Set Time Period](#)
13. [Update the Account Field of Linked Issues after a Change of the Account Field on an Epic](#)
14. [Add Tempo Planning Information](#)
15. [Select Tempo Account Automatically at Issue Creation](#)
16. [Populate Tempo Dynamic Drop-down using a REST Endpoint](#)
17. [Create a Tempo Worklog](#)
18. [Summing Tempo Worklogs with an Attribute](#)
19. [Create a Tempo Plan when an Issue is Assigned to a User](#)

So there you have it! As you can see, Jira is very flexible and there are lots of ways of setting it up. You'll find it's well-worth the effort of optimizing it and learning the tips and tricks around it.

With an enhanced Jira instance complete with automation and specialized add-ons, you make yourself more productive, efficient, and all-around happy at work.

For more information, visit tempo.io!

ABOUT TEMPO

At Tempo, a global SaaS company, we offer integrated time tracking solutions for Jira that ensure companies can apply best-in-class time management tools to drive their success. With Tempo solutions, it's easy to track time, our most constrained resource.

Companies use Tempo products to develop an aligned understanding of work and gain visibility into the true value of time. There's more to Tempo than just time tracking, though. We expanded into resource planning and budget management by creating new apps for Jira to complement our service offering.

Tempo has built a network of more than 20,000 customers, both large and small, across a range of industries all over the world. We work with more than 100 partners around the world, offering resale, training, and consultancy in local languages.



