ChatGPT

Log in

Sign up

**You said:**

jira automation for work log whenever commit is done

**ChatGPT said:**

To automate the creation of a work log in Jira whenever a commit is made in your repository, you can integrate Jira with your version control system (e.g., GitHub, GitLab, Bitbucket) and use Jira Automation.

Here’s a step-by-step guide to setting up the automation:

**Prerequisites:**

1. **Jira Cloud or Jira Server**: Ensure you have access to Jira Automation (available in Jira Cloud and Jira Server).
2. **VCS Integration**: Make sure Jira is connected to your version control system. You can connect Jira with GitHub, Bitbucket, or GitLab using the Jira integration (Jira Software has native integrations for these tools).
3. **API Access**: If you're working with a custom system or tool, you'll need Jira’s API access to create work logs programmatically.

**Steps:**

1. **Configure the VCS Webhook (Optional if Using Jira's Native Integration)**
   * For GitHub, Bitbucket, or GitLab, you can create a webhook that notifies Jira whenever a commit is made. If you're using a native integration between Jira and your version control system (such as Bitbucket or GitHub), the webhook configuration may not be necessary.
2. **Create the Jira Automation Rule**
   * Go to **Jira Settings** → **System** → **Automation Rules** (or navigate to **Project Settings** → **Automation** for a specific project).
   * Click **Create Rule**.
3. **Choose a Trigger**
   * Select the trigger for your rule. Since we are looking for a commit, select **"Issue Updated"** or **"Git Commit"** (if the integration is set up).
   * For **GitHub/Bitbucket**, you might need to use **"Commit Comment Added"** or a custom webhook to listen to commits.
4. **Set Condition**
   * You may want to add a condition like "Commit contains Jira issue key" to make sure only relevant commits to a specific issue are logged.
5. **Add Action: Log Work**
   * In the action section, select **Log Work**.
   * Configure the **Time Spent** field (this could be static, or you could use the commit message to infer work time).
   * You can also set other fields like **Work Description** to include commit details, such as the commit message.
6. **Configure Optional Actions**
   * You can also add actions like sending notifications to team members or updating issue statuses based on the commit.
7. **Save and Test**
   * After configuring the rule, save it, and test the automation by making a commit in your repository that references a Jira issue.

**Example Automation Rule**

* **Trigger**: GitHub commit with a message containing the issue key (e.g., PROJ-123).
* **Condition**: Ensure that the commit message has a valid Jira issue key.
* **Action**: Log work on the Jira issue related to the commit, using the time spent from the commit message or an automatic value.

**Advanced Custom Integration (For Custom Systems)**

If you're using a custom system or a repository that doesn’t have native Jira integration, you can achieve this by using Jira’s REST API. Here’s how:

1. **Listen for Commit Events in Your VCS**: Set up a webhook in your VCS to listen for commit events.
2. **Parse the Commit Message**: Extract the Jira issue key and the work time from the commit message.
3. **Create Work Log via Jira API**: Make an API call to Jira’s /rest/api/2/issue/{issueIdOrKey}/worklog endpoint.

**API Request Example:**

bash

CopyEdit

POST /rest/api/2/issue/{issueIdOrKey}/worklog

Authorization: Bearer <your-jira-api-token>

Content-Type: application/json

{

"timeSpentSeconds": 3600,

"comment": "Commit made for feature implementation",

"started": "2025-02-10T10:00:00.000+0000"

}

You would use the extracted Jira issue key and work time from your commit message in the API call.

By following these steps, you can automatically log work in Jira whenever a commit occurs, streamlining your workflow.

**You said:**

where to add a condition like "Commit contains Jira issue key"

**ChatGPT said:**

To add a condition like "Commit contains Jira issue key" in a Jira Automation rule, you will need to use a **Jira Automation rule** with the **trigger** related to commits, such as **"Commit Comment Added"** or **"Issue Updated"** (depending on how your VCS is integrated with Jira).

Here’s a step-by-step guide on where and how to add that specific condition:

**Steps to Add a Condition: "Commit Contains Jira Issue Key"**

1. **Go to Jira Automation**
   * Navigate to **Jira Settings** (click the gear icon in the top-right corner of Jira).
   * Select **System** → **Automation Rules** or go to **Project Settings** → **Automation** if you're working in a specific project.
2. **Create or Edit an Automation Rule**
   * Click on **Create Rule** to create a new rule or select an existing rule to edit.
3. **Select the Trigger**
   * Choose the appropriate trigger for the rule:
     + For example, choose **"Commit Comment Added"** or **"Issue Updated"** if you want to trigger the automation based on a commit message.
     + If you’re using a specific VCS integration like GitHub, Bitbucket, or GitLab, this trigger might differ.
4. **Add Condition for Jira Issue Key**
   * After selecting your trigger, click on **New Condition**.
   * Choose **"Advanced Compare Condition"** (this allows you to use a smart value to extract and check the issue key from the commit message).
5. **Set up the Condition**
   * For **Condition Type**, choose **"Advanced Compare Condition"**.
   * Set the condition to check if the commit message contains a Jira issue key:
     + **First value**: {{commit.message}} (this refers to the commit message).
     + **Condition**: matches regular expression.
     + **Second value**: ([A-Z]+-\d+) (this is a regular expression that matches Jira issue keys like PROJ-123).

The regular expression ([A-Z]+-\d+) matches any string that looks like a Jira issue key (e.g., ABC-123, PROJ-456).

This ensures the automation only triggers when the commit message includes a valid Jira issue key.

1. **Add Actions**
   * After the condition is added, you can proceed to add your desired actions, like **Log Work**, **Transition Issue**, etc.

For example, you could use the **Log Work** action to log the time spent based on the commit.

1. **Save and Test the Rule**
   * Save the automation rule and test it by making a commit that includes a Jira issue key in the commit message (e.g., "Fixed bug in PROJ-123").
   * Check if the rule correctly logs work or performs other actions based on your setup.

**Example Rule Configuration**

* **Trigger**: Commit Comment Added (if the VCS is integrated with Jira).
* **Condition**:
  + **Advanced Compare Condition**:
    - **First Value**: {{commit.message}}
    - **Condition**: matches regular expression
    - **Second Value**: ([A-Z]+-\d+) (to check for issue keys).
* **Action**: Log Work or perform any action like **Transition Issue** or **Send Notification**.

**Summary**

By setting up this condition, you make sure that the automation rule only runs if the commit message contains a valid Jira issue key. This helps to ensure that your work log, transition, or other actions only apply to relevant Jira issues connected to the commit.

**You said:**

For example, you could use the Log Work action to log the time spent based on the commit. hpw can i make this action

**ChatGPT said:**

To use the **Log Work** action in Jira Automation to log time based on the commit (such as logging the time spent as mentioned in the commit message), you need to configure the rule to extract the time information from the commit message and use it in the **Log Work** action.

**Here's how you can set up this action:**

**1. Create/Edit Automation Rule**

* Go to **Jira Settings** → **System** → **Automation Rules** (or **Project Settings** → **Automation** for a specific project).
* Either create a new rule or edit an existing one.

**2. Set the Trigger**

* Choose the appropriate trigger based on your version control system (VCS) integration. For example:
  + **"Commit Comment Added"** (if your VCS is integrated with Jira).
  + **"Issue Updated"** (if the commit is updating an issue).

**3. Add Condition to Ensure Commit Contains Jira Issue Key (Optional but Recommended)**

* Add a condition to make sure the commit message contains a Jira issue key, as previously explained.
* **Condition**: Use **Advanced Compare Condition** and check if the commit message contains a Jira issue key (e.g., PROJ-123).

**4. Add Action: Log Work**

* After setting up the trigger and condition, you can now add the **Log Work** action.

Here’s how:

1. **Click on "New Action"**.
2. Select **"Log Work"** from the available actions.

**5. Configure the Log Work Action**

In the **Log Work** action, you will need to configure several fields. Since you're aiming to log work based on the commit, you'll likely want to extract the **time spent** from the commit message.

* **Time Spent**: This is the most important field, as it represents the time you want to log. If the commit message contains the time spent (for example, 3h or 1d 2h), you can extract it using smart values.
  + For example, if the commit message contains 3h (3 hours), use the smart value {{commit.message}} to extract the time information.

**Example:**

* + In the **Time Spent** field, you can enter {{commit.message.match("(\d+)(h|m|d)").first}}. This regular expression would capture the first occurrence of time duration like 3h, 2d, etc.

Or you can parse the commit message in a more advanced way to extract both the issue key and time, depending on the format you're using in your commits.

* **Work Description**: You can include a description or message related to the commit. This is optional, but it’s helpful to understand the context of the work logged.
  + Example: Log work based on commit: {{commit.message}}.
* **Started**: You can optionally add a timestamp for when the work was done. If you want to log the time with the actual commit time, use a smart value like {{commit.timestamp}} (if available).
  + Example: {{commit.timestamp}}.
* **Remaining Estimate**: If you need to update the remaining estimate, you can set this field. However, this is typically used in combination with time tracking tools (optional for your use case).

**6. Save the Rule and Test**

* After configuring the **Log Work** action, save the automation rule.
* Test the rule by making a commit that includes an issue key and a time estimate (for example, PROJ-123 fixed bug, 2h in the commit message).
* Check if the work is logged automatically in the related Jira issue.

**Example Setup: Logging Work Based on Commit**

Let's walk through an example where the commit message contains both the Jira issue key and the time spent (e.g., "PROJ-123 Fixed bug, 2h").

1. **Trigger**: Use **"Commit Comment Added"** (triggered by a commit comment).
2. **Condition**: Add a **"Advanced Compare Condition"** to check if the commit message contains a Jira issue key (e.g., ([A-Z]+-\d+)).
3. **Action**: **Log Work**
   * **Time Spent**: {{commit.message.match("(\d+)(h|m|d)").first}} (captures time from commit, e.g., "2h").
   * **Work Description**: Fixed issue based on commit message: {{commit.message}}.
   * **Started**: {{commit.timestamp}} (this logs the time the commit was made).

**Notes:**

* If your commit message follows a specific format (e.g., ISSUE\_KEY fixed bug in 2h), adjust the regular expression to match that format.
* The time format in Jira can be hours (h), minutes (m), and days (d), so make sure the smart value is able to correctly interpret this.
* Test thoroughly to make sure that the work is logged correctly, and the commit message format matches what you expect.

This automation will log the time spent on the Jira issue whenever a commit is made with the correct Jira issue key and time spent.

**You said:**

this is not working

**ChatGPT said:**

I understand that the automation might not be working as expected. Let's go through a more precise and practical approach to logging work in Jira based on commit messages.

Since **commit messages** are often stored as **text** and **time** can be tricky to parse, there are a few important steps and considerations that can help make this work smoothly.

Here’s a revised approach using **Jira Automation** and **commit messages** with a simple working example:

**1. Jira Automation Rule Setup**

First, let's ensure we have the proper foundation for your automation rule.

**Trigger: "Commit Comment Added"**

1. Go to **Project Settings** → **Automation** (or **System** → **Automation Rules** for global rules).
2. Create a new rule with **"Commit Comment Added"** as the trigger (this requires integration between Jira and your VCS).
   * **Trigger**: "Commit Comment Added" will trigger the automation whenever a comment is added to a commit.
   * In the **Condition** section, ensure that you're verifying that the commit comment contains the Jira issue key (e.g., PROJ-123).

**2. Condition: Verify Commit Contains Jira Issue Key**

If you want to ensure the commit is tied to a valid Jira issue key (e.g., PROJ-123), you can use an **Advanced Compare Condition**.

* Choose **Advanced Compare Condition**.
* Set it up as follows:
  + **First value**: {{commit.message}} (the commit message).
  + **Condition**: matches regular expression.
  + **Second value**: ([A-Z]+-\d+) (this matches typical Jira issue keys like PROJ-123).
* This ensures that the commit message contains a valid Jira issue key.

**3. Action: Log Work Based on Commit Message**

Now, we’ll focus on the **Log Work** action, where you want to log the time spent based on the commit message.

**Key Consideration: Commit Time Format**

* **Time Parsing**: If the commit message contains time, we need to match it correctly. Let’s assume that your commit message has the format:
  + **<issue-key> <description> <time>** (e.g., PROJ-123 Fixed bug in 2h).

**Action: Set up the "Log Work" action**

1. **Log Work Action**:
   * Click on **New Action** → **Log Work**.
2. **Time Spent**:
   * Here’s where we extract the time from the commit message.
   * Use the smart value {{commit.message}} to refer to the commit message.
   * Use a regular expression to extract time like 2h, 1d 30m, etc.

**For example**:

* + - In the **Time Spent** field, use:

plaintext

CopyEdit

{{commit.message.match("(\d+)(h|m|d)").first}}

This regular expression will capture the first occurrence of a time format such as 2h, 1d, or 30m in the commit message.

* + - For example, if the commit message is PROJ-123 Fixed bug in 2h, this regular expression would capture 2h and log it as time spent.
  + **Work Description**: Optionally, you can add the commit message or a custom description to describe the work logged.

plaintext

CopyEdit

Fixed issue based on commit: {{commit.message}}

* + **Started**: You can use {{commit.timestamp}} if you'd like the work to be logged at the time of the commit, or leave this field blank to log it at the current time.

1. **Save the Rule** and test it by making a commit that follows the format PROJ-123 Fixed bug in 2h.

**4. Common Issues and Troubleshooting**

If this still doesn't work, here are some potential issues and steps to resolve them:

* **No Time in Commit Message**: If the commit message doesn’t contain time (e.g., 2h or 1d), the rule won’t know how to log time. You must ensure that the commit message follows a consistent format (e.g., PROJ-123 Fixed bug in 2h).
* **Incorrect Regular Expression**: If the regular expression doesn't match the time format you're using in your commit message, the rule won't log any time. Make sure the regex matches your expected format for time (e.g., hours (h), minutes (m), or days (d)).
* **Incorrect Smart Values**: If the smart value {{commit.message}} doesn’t correctly reference the commit message, you can check whether your version control system integration properly supports commit data in Jira. You might need to adjust the smart value to fit the integration (e.g., {{commit.comment}} or {{commit.description}}).

**Example of Commit Message Format:**

* Commit message: PROJ-123 Fixed bug in 2h
  + This commit would match the Jira issue key (PROJ-123) and extract the time 2h based on the regular expression ({{commit.message.match("(\d+)(h|m|d)").first}}).

**Conclusion**

* The core of this solution is making sure the **commit message contains time in a recognizable format** (e.g., 2h, 1d 30m).
* Use the **Log Work** action in Jira Automation, and ensure you are correctly extracting time using a regular expression from the commit message.
* If it's still not working, you may want to test with a simpler commit message and confirm whether the commit information is correctly passed to Jira.

If the problem persists, let me know what error or behavior you're experiencing, and I’ll be happy to help troubleshoot further!

**You said:**

{{smart values}} condition 02/10/2025, 15:28:08 The following issues did not match the condition: TES-3

**ChatGPT said:**

It seems like you're encountering an issue with the condition where the rule is not matching the expected Jira issue key (TES-3), and it's possible that the smart value you're using isn't correctly referencing the commit message or the issue key in the correct format.

Let’s address this issue step by step:

**Key Points:**

* **The smart value you're using** might not be correctly capturing the issue key.
* **The condition may not be set properly** to match the Jira issue key from the commit message.

Let’s break it down and make sure everything is set up properly:

**1. Trigger: Ensure the Correct Commit Trigger**

First, double-check that the **commit trigger** is correctly configured.

* Trigger: **"Commit Comment Added"** (if you are using a commit comment).
* Ensure the commit message contains the Jira issue key, like TES-3.

**2. Condition: Matching the Issue Key in the Commit Message**

You want to check that the **commit message contains the Jira issue key**, so you’ll need to use a **regular expression** in the **condition** to match the format of a Jira issue key (e.g., TES-3).

To do that, follow these steps:

1. **Set Condition**:
   * Use the **Advanced Compare Condition** to check if the commit message contains the Jira issue key.
2. **Condition Setup**:
   * **First value**: {{commit.message}} (this will be the commit message).
   * **Condition**: matches regular expression.
   * **Second value**: ([A-Z]+-\d+) (this regular expression will match any typical Jira issue key format like TES-3 or PROJ-123).

**Explanation of Regular Expression**:

* + ([A-Z]+-\d+): This regular expression matches:
    - [A-Z]+: One or more uppercase letters (for the project key, e.g., TES).
    - -: A hyphen separator.
    - \d+: One or more digits (for the issue number, e.g., 3).

So the commit message TES-3 Fixed bug in the system should match the regular expression TES-3.

**3. Troubleshooting the Condition**

If the condition still doesn’t match, here are some things to check:

1. **Check the Commit Message**: Ensure the commit message is properly formatted and includes the issue key (e.g., TES-3). If the commit message doesn’t contain the issue key in the expected format, the condition will fail.
2. **Check the Smart Value**: If you're using a smart value like {{commit.message}}, confirm that it's correctly extracting the message from the commit. If the smart value isn’t working, it could be that your VCS integration with Jira is not passing the commit message correctly.
   * If needed, you can check the **Audit Log** in Jira Automation to see the exact data that’s being passed into the rule. This can help identify whether the smart value is returning the correct commit message.

**4. Test with a Simple Condition**

To isolate the issue, try testing a simpler condition first:

* **Condition**: Check if the commit message contains a specific string (e.g., TES-3), rather than using a regular expression.
  + **First value**: {{commit.message}}
  + **Condition**: contains
  + **Second value**: TES-3

This test will check if the commit message explicitly contains TES-3. If this works, we know that the commit message is being passed correctly, and we can move back to the regular expression.

**Example Automation Setup:**

* **Trigger**: "Commit Comment Added".
* **Condition**: **Advanced Compare Condition**.
  + **First value**: {{commit.message}}
  + **Condition**: matches regular expression.
  + **Second value**: ([A-Z]+-\d+) (matches issue key format like TES-3).

**5. Final Debugging Steps**

If this is still not working, here are some things to consider:

* **Double-check the webhook configuration**: Ensure that the commit comments are properly sent to Jira and that the data contains the full commit message.
* **Test using simple commit messages**: Try using a simpler commit message (e.g., TES-3 Fixed issue) to test the condition.
* **Check Audit Logs**: In Jira Automation, go to the **Audit Logs** to see if the automation rule is being triggered and which data is being passed.

**Conclusion**

By setting the condition to **match the regular expression** ([A-Z]+-\d+) on {{commit.message}}, you should be able to successfully match Jira issue keys like TES-3 in your commit message. If this still doesn't work, check the **audit logs** and ensure that the commit messages are passed correctly from your VCS to Jira.

Let me know if this helps or if you run into any specific errors during testing!

**You said:**

{{smart values}} condition 02/10/2025, 15:28:08 The following issues did not match the condition: TES-3 , TES-x is issue id based in project, how to generalise

**ChatGPT said:**

Ah, I see now! You are looking to generalize the condition so it works with **any issue key format**, not just specific issue IDs. The smart value condition you are using might be working for a specific format, but you need it to work for **any issue key** (e.g., TES-3, TES-10, TES-XYZ, etc.) and possibly across multiple projects.

**Key Understanding:**

You want a condition that matches **any Jira issue key** format, which is generally:

* **Format**: PROJECTKEY-123 (e.g., TES-3, TES-101, PROJ-45).
* **PROJECTKEY**: One or more uppercase letters.
* **ISSUE ID**: A number that follows the hyphen (-).

**Solution: Generalizing the Condition for Any Jira Issue Key**

You can use a **regular expression** to match **any valid Jira issue key** format, regardless of the project key (e.g., TES, PROJ, or any other project key).

**Regular Expression for Jira Issue Key**

The regular expression to match **any valid Jira issue key** would be:

r

CopyEdit

([A-Z]+-\d+)

This will match:

* **[A-Z]+**: One or more uppercase letters (representing the project key, e.g., TES or PROJ).
* **-**: The hyphen separator.
* **\d+**: One or more digits (representing the issue number, e.g., 3, 123, etc.).

This regular expression will match any string like:

* TES-3
* TES-10
* PROJ-100
* ABC-123

**How to Implement in Jira Automation**

1. **Condition**: Use an **Advanced Compare Condition** to check if the commit message contains a Jira issue key.
2. **Setup for Generalization**:
   * **First Value**: {{commit.message}} (this refers to the commit message).
   * **Condition**: matches regular expression.
   * **Second Value**: ([A-Z]+-\d+) (this regular expression will match any valid Jira issue key format like TES-3, PROJ-100, etc.).

**Full Setup Example:**

**Step 1: Trigger**

* **Trigger**: "Commit Comment Added" (this depends on your VCS integration with Jira).

**Step 2: Condition (Advanced Compare Condition)**

* **First Value**: {{commit.message}}
* **Condition**: matches regular expression
* **Second Value**: ([A-Z]+-\d+) (this matches any Jira issue key).

**Step 3: Action (e.g., Log Work)**

If the commit message contains a valid Jira issue key, you can add an action like **Log Work** or any other relevant action.

**Example Commit Message:**

For instance, a commit message like:

perl

CopyEdit

TES-3 Fixed bug in the system

will match the regular expression ([A-Z]+-\d+), and Jira will recognize it as a valid issue key.

**Example Rule Configuration:**

1. **Trigger**:
   * Commit Comment Added
2. **Condition**:
   * **First Value**: {{commit.message}}
   * **Condition**: matches regular expression
   * **Second Value**: ([A-Z]+-\d+)
3. **Action**:
   * **Log Work** (or any other action you want to take).

**Testing:**

* If your commit message is TES-3 Fixed bug, it will match TES-3.
* If the message is PROJ-45 Refactor code, it will match PROJ-45.
* It works with any project key (e.g., TES, PROJ, ABC) and any valid issue number (e.g., 1, 10, 123).

**Troubleshooting:**

* If you're still facing issues, ensure the commit message **actually contains the issue key** in the expected format.
* Review the **Audit Log** in Jira Automation to see exactly what values are being passed into the condition. Sometimes, the smart value might not contain what you expect.

**Conclusion:**

By using the regular expression ([A-Z]+-\d+) in the **Advanced Compare Condition**, you ensure that your automation rule will match any valid Jira issue key in the commit message, whether it's TES-3, PROJ-101, or any other format.

Let me know if you need further adjustments or encounter additional issues!

Top of Form

Get smarter responses, upload files and images, and more.

Log in

Sign up



Attach

Search

Reason

Voice

Bottom of Form

ChatGPT can make mistakes. Check important info.

?