

Integration Manual

Jira

How to integrate cloud-based Jira implementations

Document Information

Code: **IM-JIRA**

Version: **2.0**

Date: **17 April 2025**

Copyright © 2025 Admin By Request

All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement or nondisclosure agreement (NDA). The software may be used or copied only in accordance with the terms of those agreements.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the customer's stated use without the written permission of Admin By Request.

Contact Admin By Request



+64 21 023 57020



marketing@adminbyrequest.com



adminbyrequest.com



Unit C, 21-23 Elliot St, Papakura, NZ

Table of Contents

Overview	1
Introduction	1
In this document	1
Prerequisites	2
Integration Tasks	5
1. Creating custom fields	5
2. Creating a custom Request Type	8
3. Integrating Jira with Admin By Request	12
Updating Jira issues	16
What next?	16
Creating Jira automation rules	17
Using the Jira Audit log	17
Linking back to the ABR Portal	17
Example 1 - Populating issue fields after issue creation	18
Example 2 - Approving or Denying ABR Requests in Jira	21
Document History	28
Index	29

Overview

Introduction

This topic describes integration between Admin By Request and Jira for cloud-based Jira implementations. The integration is relatively simple, comprising three mechanisms: one to create issues in Jira from requests made in ABR, one to handle approved requests and one to handle declined requests. This is the extent of our Jira integration; it is up to the customer to further automate the updating of tickets through Jira automation rules.

Nevertheless, we include in this topic two simple automation rules to illustrate how some things might be done. Customers can use these as the basis for their own automation workflows.

IMPORTANT

- The integration covered here is available for Jira Cloud, but not on-premise installations like Jira Data Center Edition.
- Disclaimer - Admin By Request is not a Jira consultancy and we do not have extensive experience working with Atlassian tools, including Jira. Our integrations and examples are provided for the customer's convenience; they do not carry any warranty whatsoever and must be used entirely at the customer's own risk. We recommend testing everything in a non-production environment and we are not liable for any downtime or loss of productivity or data that might result from using the integrations.

In this document

["Prerequisites" on the next page](#)

["1. Creating custom fields" on page 5](#)

["2. Creating a custom Request Type" on page 8](#)

["3. Integrating Jira with Admin By Request" on page 12](#)

["Updating Jira issues" on page 16](#)

["Creating Jira automation rules" on page 17](#)

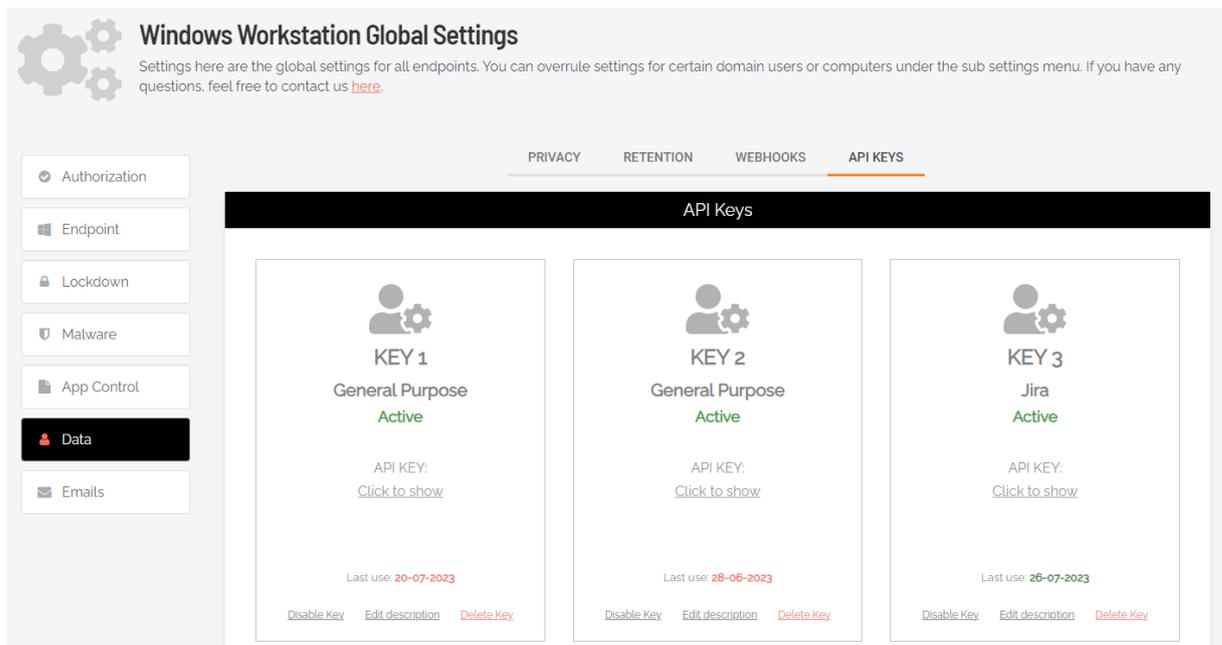
Prerequisites

To complete the setup, you will need the following:

- A. A Jira Service Management username and an Admin By Request Portal username.
You must have administrator access to both Jira Service Management and the ABR Portal.
- B. A Jira Service Management API token associated with your username.
For information on managing API tokens in Jira, refer to <https://support.atlassian.com/atlassian-account/docs/manage-api-tokens-for-your-atlassian-account/>.
- C. An API key in your ABR Portal:

Create an API key

1. Log in to the portal and navigate to **Settings > Tenant Settings > API Keys > API KEYS**.
2. To create a new API key, click button **Add New**.
3. If you want to name this key, click **Edit description** and name it accordingly.
4. Click **Save** to save the new API key:



Copy the key to the clipboard, ready for pasting in the next step.

Test the API key

The following test uses VS Code to send a curl statement to the ABR server's API interface. A successful test returns **200 OK**, plus a list of ABR requests in JSON format.

NOTE

If there are no ABR requests in *any* category (pending, approved, denied or quarantined) then the server will still return **200 OK**, but no JSON data.

To test the API Key::

1. In your portal, identify the URL to use for API requests. This is based on the data center to which you are connected.

To determine your data center, go to page [Tenant Settings > API Keys](#) in the portal and check which API prefix is shown under **About API Keys**. The data center (which is also the API prefix) will be one of the following:

- **https://dc1api.adminbyrequest.com** (Europe)
- **https://dc2api.adminbyrequest.com** (USA)

Make a note of your prefix - among other things, this is the domain used when an API Key is created.

You can also see your API prefix on the API web pages (e.g. [Public API > Auditlog API](#)). However, a small script runs in the background that determines to which data center you are attached, so JavaScript must be enabled in your browser for this to work.

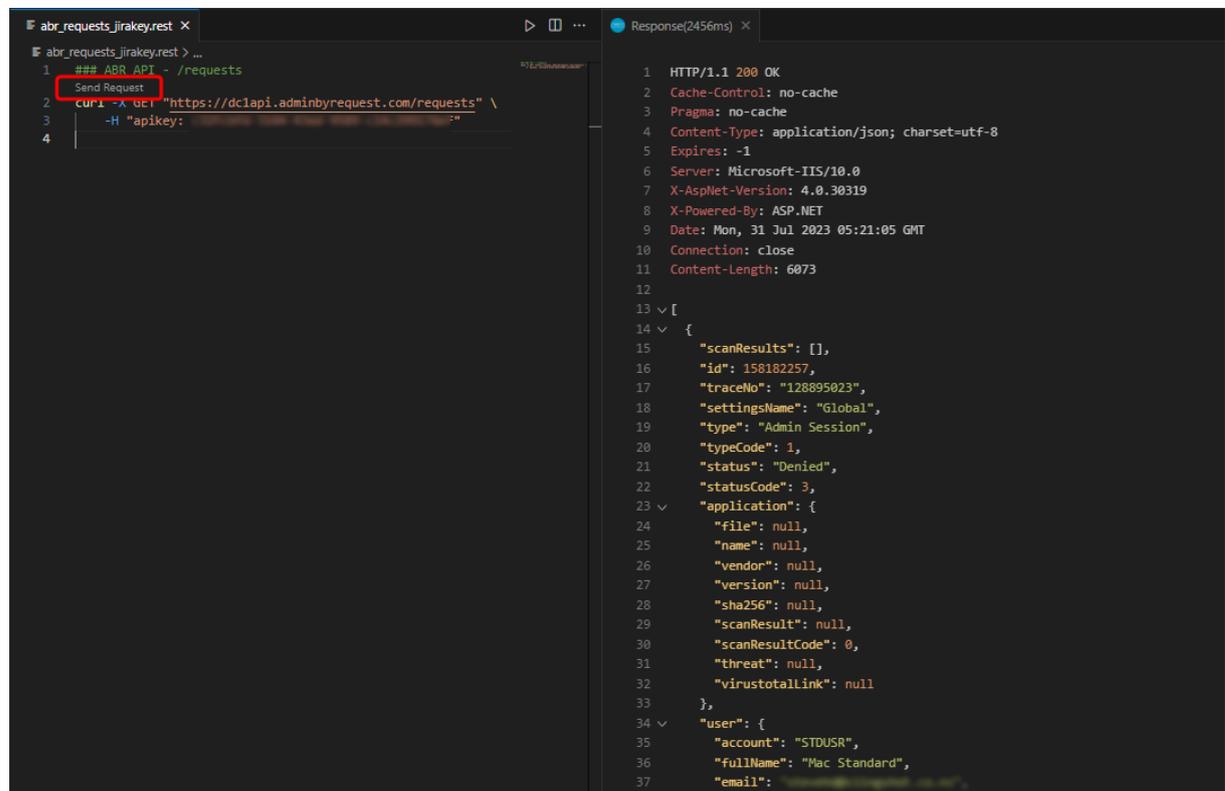
The example below uses **https://dc1api.adminbyrequest.com/requests** as the URL.

2. Enter the following curl statement, using your URL and your API key:

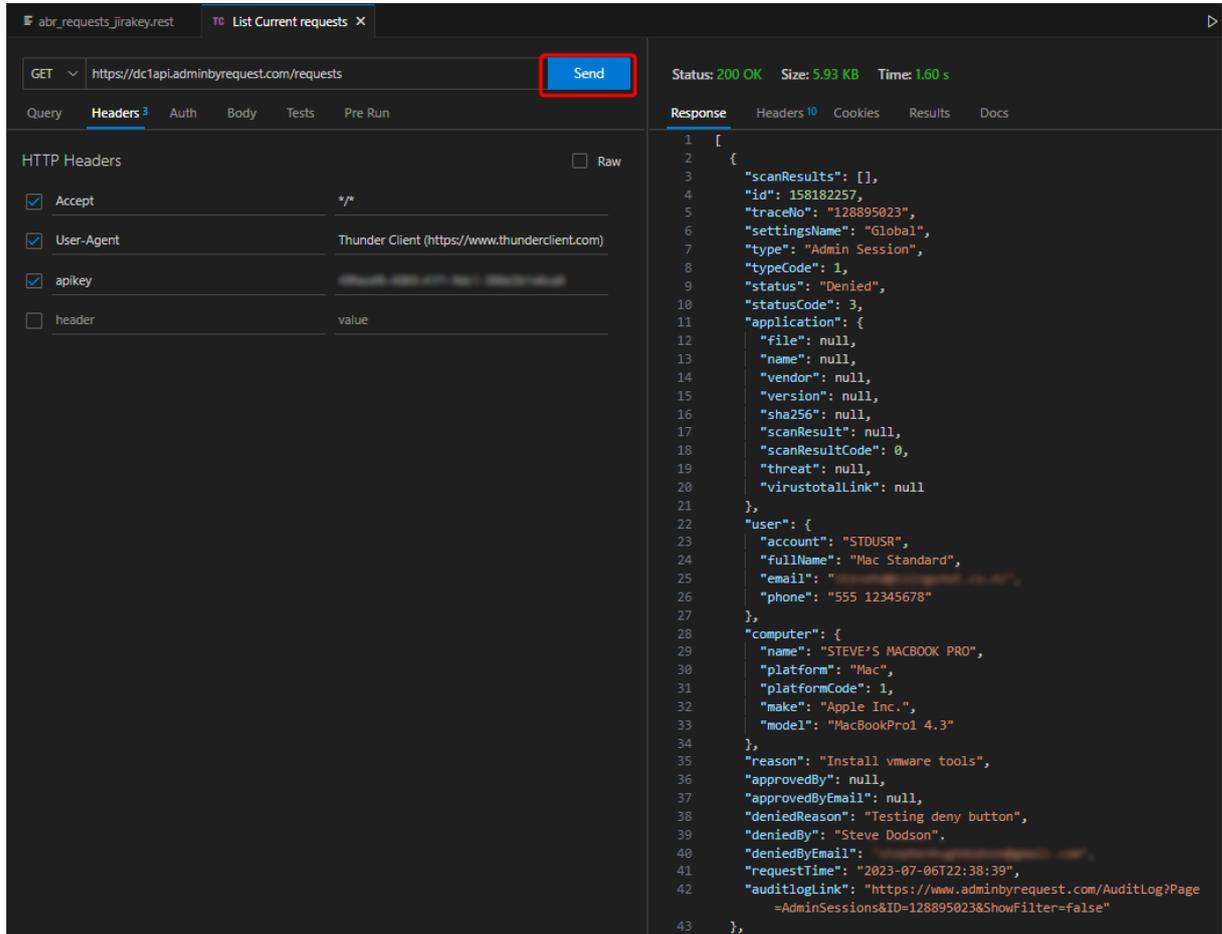
```
curl -X GET "https://dc1api.adminbyrequest.com/requests" \
-H "apikey: <your API key>"
```

3. Send the curl statement. The following examples show the response in VS Code from the ABR server using firstly the REST client and secondly the Thunder client.

REST client



Thunder client



There are multiple ABR requests returned across all states (pending, approved, denied, etc.) - only the first request is shown. Scroll the response you get back to see all requests.

Once you have this information, we're ready to get started.

There are three main tasks required to configure Jira integration. The next chapter describes each task in detail.

Integration Tasks

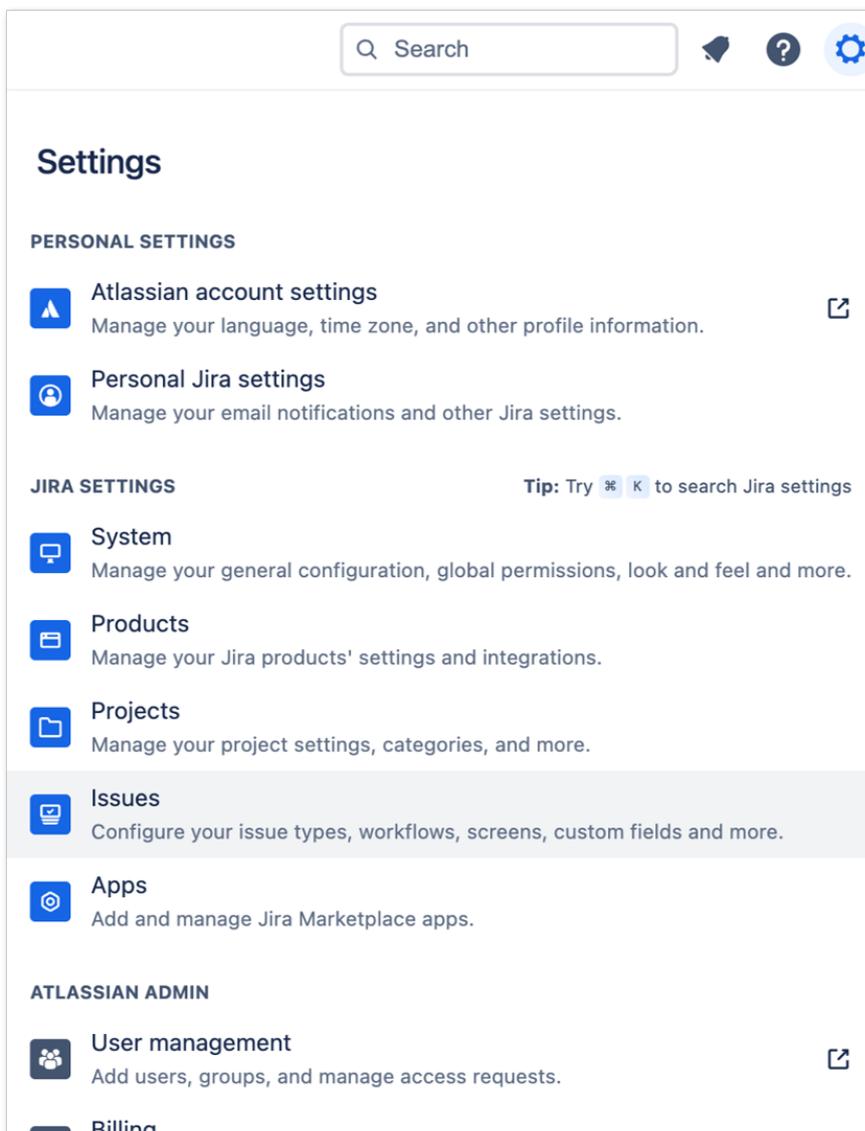
1. Creating custom fields

The integration requires you to set up a few custom fields. These fields hold information about:

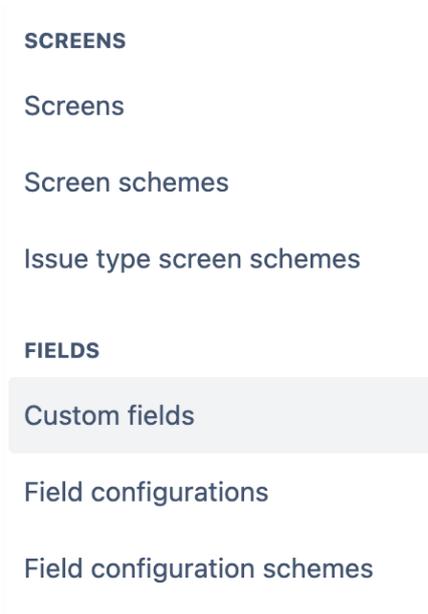
- The ID of the request from Admin By Request (can be used for further automation).
- The name of the approver (if the request is approved from the platform or via another integration).
- The reason supplied if a request is denied.

Create custom fields

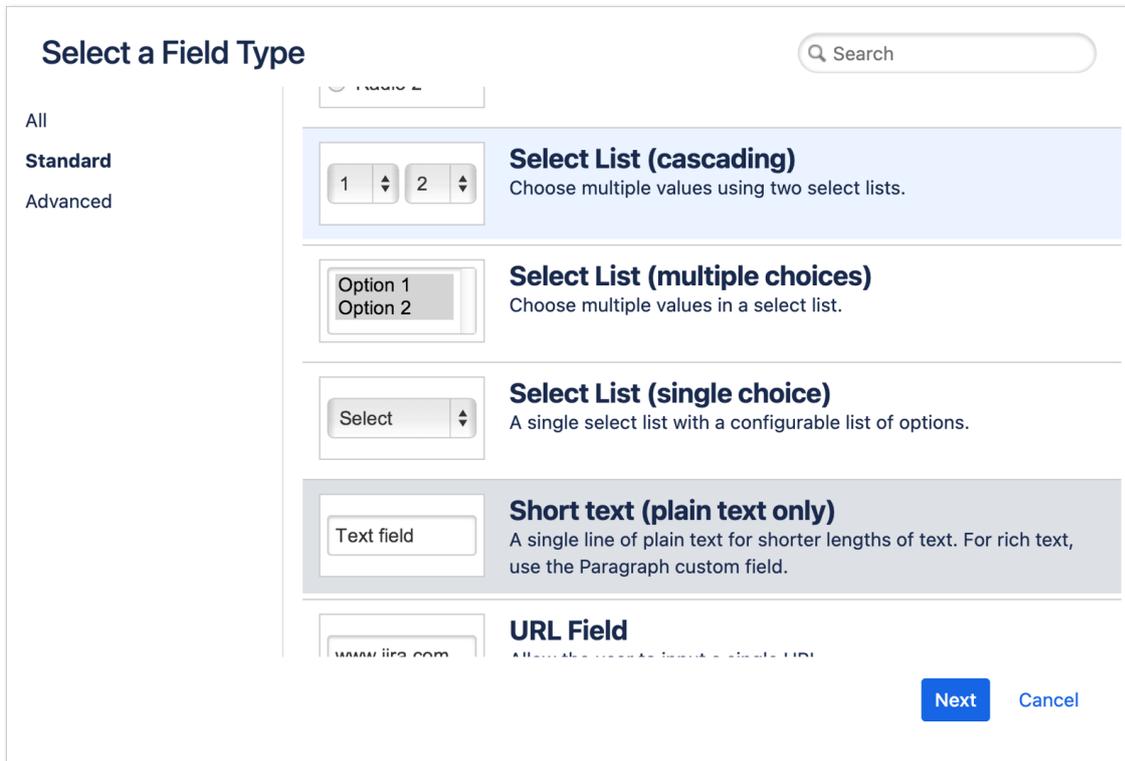
1. Log in to Jira Service Management and click the cog in the upper right corner to navigate to **Settings > Issues**:



2. In the left menu, select **Custom fields**:



3. In the *Custom fields* screen, click button **Create custom field** and select Field Type **Short text (plain text only)**:



4. Name the field (e.g. **ABR Request ID**) and, optionally, provide a description.
5. Repeat these steps for field **ABR Handled by**.

- Repeat the steps one more time for field **ABR Reason**, but select the multiline option **Paragraph (supports rich text)** as the Field Type.

NOTE

You don't have to select any specific views for the fields - these will be assigned when creating the *Request Type* in the next section.

- Finally, find the ids of the fields just created. To do this, make sure at least one issue exists in your project. Identify the issue number and enter the following URL in a browser:

```
https://<your-jira-website>.atlassian.net/rest/api/2/issue/<your-jira-issue-number>?expand=names
```

Copy and paste the resulting JSON into an editor that can format it nicely (e.g. Notepad++) and find the labels of the fields created.

The following example has issue number **ABRJIRA-40** (from the "key" field) and shows both field id (**customfield_10064**) and field name for **ABR Request ID**:

```

4      "self": "https://abr-integration-test.atlassian.net/rest/api/2/issue/10176",
5      "key": "ABRJIRA-40",
6      "names": {
7          "statuscategorychangedate": "Status Category Changed",
8          "fixVersions": "Fix versions",
9          "resolution": "Resolution",
10         "lastViewed": "Last Viewed",
11         "customfield_10061": "Category",
12         "customfield_10062": "Atlas project key",
13         "customfield_10063": "Atlas project status",
14         "customfield_10064": "ABR Request ID",
15         "customfield_10066": "ABR Handled by",
16         "priority": "Priority",
17         "customfield_10067": "ABR Reason",
18         "customfield_10068": "Time to done",
19         "labels": "Labels",
20         "aggregatetimeoriginalestimate": "Σ Original Estimate",
21         "timeestimate": "Remaining Estimate",
22         "versions": "Affects versions",
23         "issuelinks": "Linked Issues",
24         "assignee": "Assignee",
25         "status": "Status",
26         "components": "Components",

```

The field id for ABR Request ID is needed later in task ["Overview" on page 1](#).

NOTE

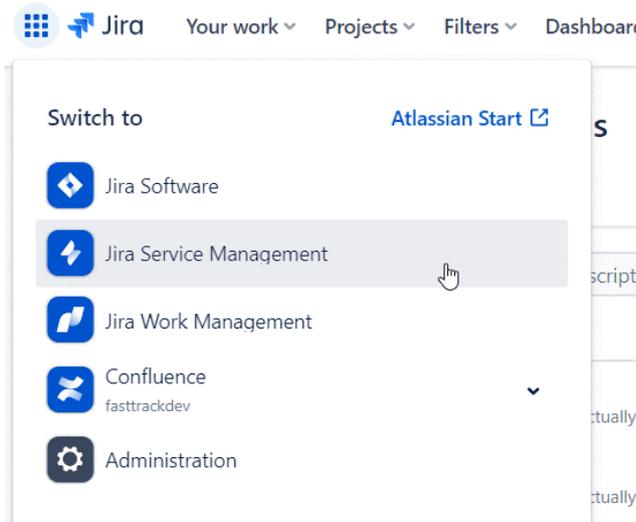
This is an example - your field id(s) will almost certainly be different.

2. Creating a custom Request Type

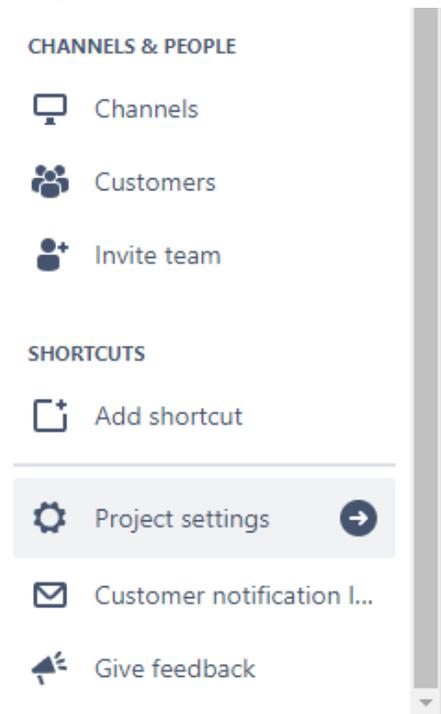
A custom Request Type enables Jira to identify any requests coming from the Admin By Request servers.

Create a custom Request Type

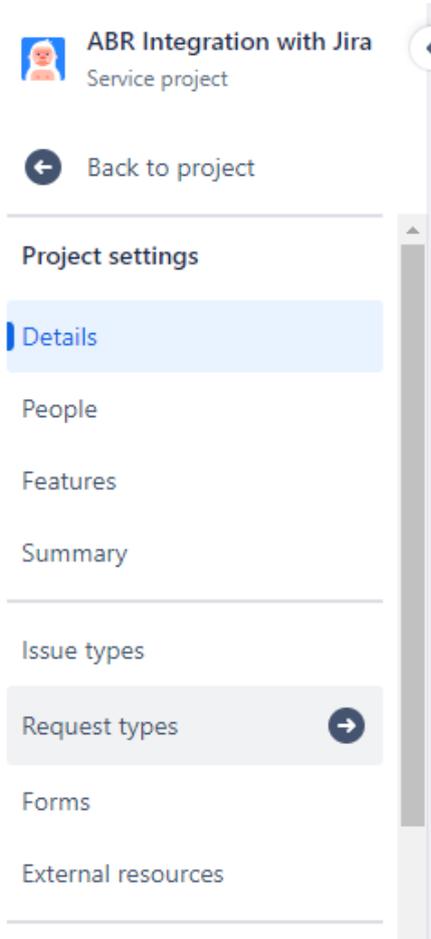
1. Login to Jira and, from the Switch to... menu, select Jira Service Management:



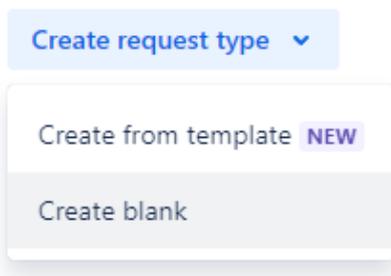
2. From within Jira Service Management, either create a new project or select the project you want to integrate with. Open the project and select **Project settings** towards the bottom of the left menu:



- From the Project settings menu, select **Request types**:



- On the *Service requests* screen, use the **Create request type** button (top right) to create a request type (click the button and select **Create blank**):



- Give the request type a name and select the Issue type as either:
 - [System] Service request**
 This sends requests to Jira and updates them as they get approved or denied, OR
 - [System] Service request with approvals**
 Works in the same way as the ordinary service request type, but allows designated approvers to approve or deny requests from within Jira.
 With a bit of automation, this also enables Jira to call the Admin By RequestAPI to approve or deny requests directly from within Jira.

NOTE

The automation example ("[Overview](#)" on page 1) requires that **[System] Service request with approvals** is the option chosen at this point:

New service request type [X]

Name *
Requests from Admin By Request

Description ⓘ
[Empty text area]

Icon
[+ Change icon]

Issue type ⓘ
[System] Service request [v]

HAS ALL REQUIRED SERVICE REQUEST FIELDS

- Task
- [System] Service request
- [System] Incident
- [System] Problem
- [System] Change
- [System] Post-incident review
- [System] Service request with approvals

6. Select the portal group to assign to this new request type (or leave all options blank):

Select a portal group [X]

To hide the request type from your help center, leave all groups unselected.

- Common Requests
- Computers
- Logins and Accounts
- Applications
- Servers and Infrastructure

+ Create group Cancel Back Create

- Finally, drag and drop the three newly created custom fields, as well as the **Description** field into the view of the new request type:

The screenshot shows the configuration page for a new request type. At the top, there are three tabs: 'Request form' (highlighted with a red box), 'Issue view', and 'Workflow statuses'. Below the tabs is a header 'Requests from Admin By Request' with a plus icon. Underneath, there is a text input field for 'Request type description' with the placeholder 'Enter text'. A dashed box contains a list of fields: 'Instructions', 'Summary' (with a 'REQUIRED' label), 'Description', 'ABR Handled by', 'ABR Reason', and 'ABR Request ID' (with a 'HIDDEN' label). A red box highlights the 'Description', 'ABR Handled by', 'ABR Reason', and 'ABR Request ID' fields. At the bottom of the dashed box, there is a 'Forms' section with the text 'Attach an existing form to this request type, or create a new form using a template.' and an 'Attach form' button with a dropdown arrow.

NOTE

- Make sure you are on the **Request form** tab and not *Issue view* or *Workflow statuses*.
- Make sure the **Description** field is added along with the three custom fields. *Summary* and *Instructions* should already be there.
- You can leave the ABR Request ID field as "Use preset value and hide from portal" if you do not wish the ABR Request ID to be visible.

- Save the changes.
Everything is now ready to create the ABR-Jira integration.

3. Integrating Jira with Admin By Request

Now that the pre-requisites are in place, head over to <https://jira.adminbyrequest.com> to start the setup process.

Setup Jira integration

1. From the *Set up Jira integration* screen, check once again that you've completed all the prerequisite steps and click button **Start setup**:

 Admin By Request

Set up Jira integration

With this integration, request from Admin By Request will be synced to your Jira Service Management instance.



Before getting started, please ensure that you've completed the following steps within your Jira Service Management instance:

1. Created a custom field used for storing the name of the user who has handled the request.
2. Created a custom field used for storing the reason for denying a specific request.
3. Created a custom field used for storing the request ID.
4. Created the Request Type that you would like to use with requests from Admin By Request.
5. Ensured that the custom fields are available on the view of the request type.

You can view information about the setup in more details [here](#).

[Start setup](#)

- Retrieve your Atlassian API Token as well as your Admin By Request API key and enter these into the setup form alongside your Atlassian username and instance URL:



Set up Jira integration - Information

Please fill in the information below to set up the integration with Jira.

You Atlassian username and API token can be generated from your Atlassian profile page. The Admin By Request API key can be generated from the Admin By Request portal.

Atlassian URL

The URL of you Atlassian setup (e.g. https://myorganization.atlassian.net).

Username

The username of the Atlassian user to use for API calls.

Atlassian API token

The API token to use for the API calls.

Admin By Request API key

API key generated from the Admin By Request portal.

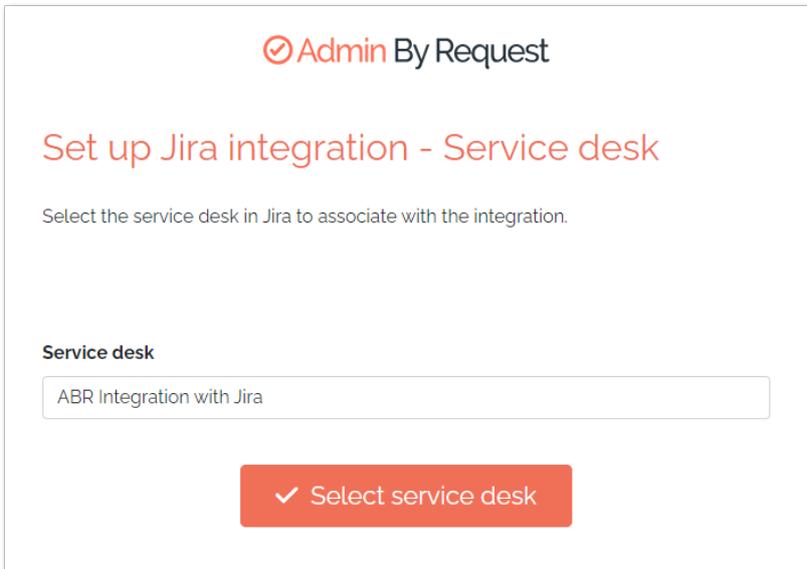
Continue →

NOTE

Your username needs to be one that is authorized to make API calls. It is generally the email address recorded under your profile at the top right of the Jira Service Management screen:



- Next, you are prompted to select the Jira service desk to associate with the integration:



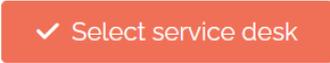


Set up Jira integration - Service desk

Select the service desk in Jira to associate with the integration.

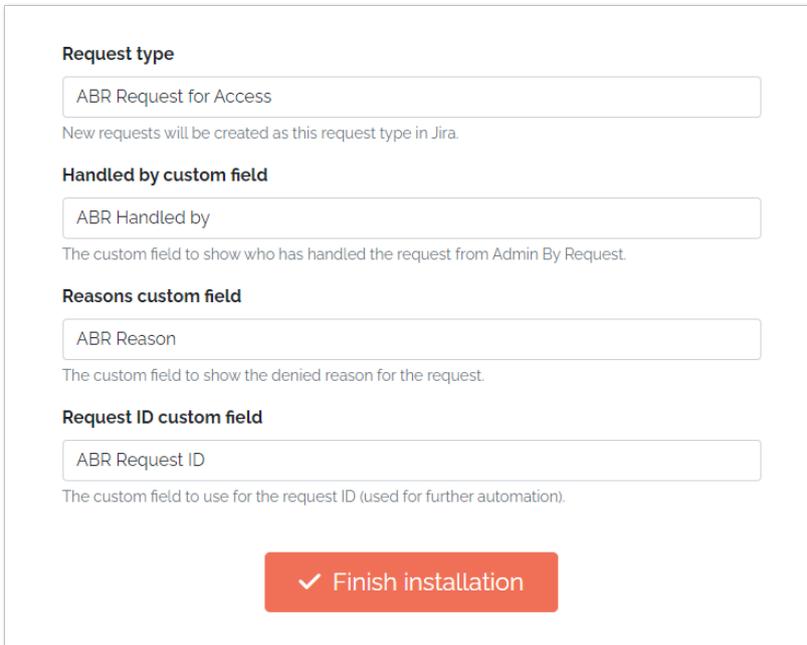
Service desk

ABR Integration with Jira



If you have more than one service desk project in Jira, a drop-down arrow will appear at the right of the *Service desk* field, allowing you to select the one you want.

- The final screen is where the custom fields you created are associated with the three required by the ABR integration. *Request type* is already selected for you - choose the other three to match your custom fields:



Request type

ABR Request for Access

New requests will be created as this request type in Jira.

Handled by custom field

ABR Handled by

The custom field to show who has handled the request from Admin By Request.

Reasons custom field

ABR Reason

The custom field to show the denied reason for the request.

Request ID custom field

ABR Request ID

The custom field to use for the request ID (used for further automation).



- Click **Finish installation** to complete setting up the Jira integration. New requests from Admin By Request will now be sent to your Jira instance as service requests.

You can check the status of your Jira integration via the portal (**Settings > Integrations > Jira**):

Jira Integrations



Integrations

Name	Functional	
abr-integration-test.atlassian.net	<input checked="" type="checkbox"/>	Delete

[New Jira Integration](#)



About Jira Integrations

The Jira integration will allow your team to interact with Admin By Request directly via Jira – giving access to features like:

- Receive incoming requests directly in Jira.
- Approve and deny requests directly from Jira.

Refer to [this page](#) for documentation.

Updating Jira issues

When a user makes a request via ABR for either an "Admin Session" or to execute something "Run As Admin", a Jira issue is created in your selected service desk instance.

NOTE

It can take several minutes after the ABR request is submitted for the issue to be created in Jira.

The Jira *Summary* indicates the type of request: **Admin session** or **Run as admin**. This field is updated once the request is either APPROVED or DENIED.

For example, the following list of issues shows one "Run as admin request" waiting for approval, two "Admin session" requests approved and one admin session denied:

T	Key	Summary
	ABRJIRA-41	Run as admin request - Notepad++ : a free (GNU) source code editor
	ABRJIRA-40	APPROVED - Admin session request
	ABRJIRA-39	DENIED - Admin session request
	ABRJIRA-38	APPROVED - Admin session request

What next?

The preceding sections cover the extent of the Admin By Request Jira integration. If you want to do more with Jira issues once they have been created, you can use Jira automation to populate issues with more information or to carry out actions such as email notifications.

IMPORTANT

The examples provided in the next section are for convenience only - we do not support any Jira automation rules and they must be used entirely at the customer's own risk.

If you do decide to work with Jira automation, note the following:

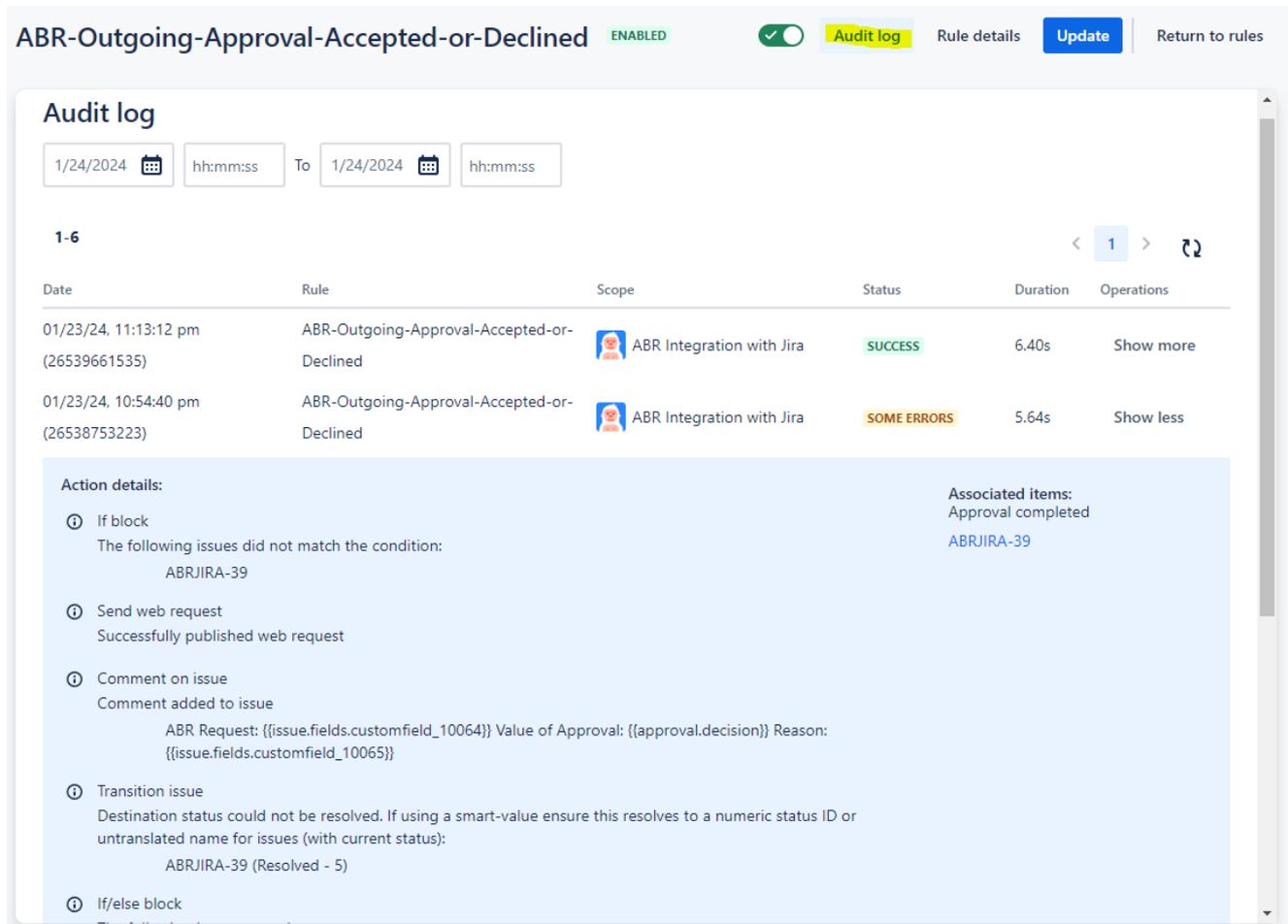
- If you selected **Service request** as the Request Type (see "[2. Creating a custom Request Type](#)" on [page 8](#)), you can use simple automation to interrogate the Summary field to carry out further actions (illustrated in "[Example 1 - Populating issue fields after issue creation](#)" on [page 18](#)).
- If you selected **Service request with Approval** as the Request Type, you can use more advanced automation, as illustrated in "[Example 2 - Approving or Denying ABR Requests in Jira](#)" on [page 21](#).

Creating Jira automation rules

Using the Jira Audit log

The Jira Audit log is a very useful tool when working with Jira automation, especially as an aid in troubleshooting.

Each rule has its own Audit Log:



ABR-Outgoing-Approval-Accepted-or-Declined ENABLED ✔ Audit log Rule details Update Return to rules

Audit log

1/24/2024 📅 hh:mm:ss To 1/24/2024 📅 hh:mm:ss

1-6 < 1 > 🔄

Date	Rule	Scope	Status	Duration	Operations
01/23/24, 11:13:12 pm (26539661535)	ABR-Outgoing-Approval-Accepted-or-Declined	 ABR Integration with Jira	SUCCESS	6.40s	Show more
01/23/24, 10:54:40 pm (26538753223)	ABR-Outgoing-Approval-Accepted-or-Declined	 ABR Integration with Jira	SOME ERRORS	5.64s	Show less

Action details:

- 📘 If block
The following issues did not match the condition:
ABRJIRA-39
- 📘 Send web request
Successfully published web request
- 📘 Comment on issue
Comment added to issue
ABR Request: {{issue.fields.customfield_10064}} Value of Approval: {{approval.decision}} Reason: {{issue.fields.customfield_10065}}
- 📘 Transition issue
Destination status could not be resolved. If using a smart-value ensure this resolves to a numeric status ID or untranslated name for issues (with current status):
ABRJIRA-39 (Resolved - 5)
- 📘 If/else block

Associated items:
Approval completed
[ABRJIRA-39](#)

Linking back to the ABR Portal

A future release of the Jira integration will almost certainly include a link back to the ABR Portal "Requests" page, from where requests can be approved or denied. Until this is available, *Example 1* below provides a workaround that customers can implement using a Jira rule (Action 1 - set fields, Description).

Example 1 - Populating issue fields after issue creation

Issue created from ABR ENABLED ✓ Audit log Rule details Update Return to rules

+ When: Issue created
Rule is run when an issue is created.

↔ Request Type equals
ABR Request for Access

✎ Then: Edit issue fields
Description, Reporter, Approvers

🔄 And: Transition the issue to
WAITING FOR APPROVAL

✉ And: Send email
myemailaddress@mycompany.com
Issue {{issue.key}} just created and waiting for approval

+ Add component

Rule details

Name *

Description

Scope

Owner *

The owner will receive emails when the rule fails.

Actor *

Actions defined in this rule will be performed by the user selected as the actor. [Learn more about rule actors in automation.](#)

Notify on error

Who can edit this rule? *

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.

1. Trigger:

+ When: Issue created
Rule is run when an issue is created.

↔ Request Type equals
ABR Request for Access

+ Issue created 🗑️

Rule is run when an issue is created. This trigger needs no configuration.

2. IF Request Type is **ABR Request for Access**:

The screenshot shows a rule configuration interface. On the left, a vertical flow of components is shown: 'When: Issue created', 'Request Type equals' (highlighted with a blue border), 'Then: Edit issue fields', and 'And: Transition the issue to WAITING FOR APPROVAL'. On the right, the 'Issue fields condition' configuration panel is open, showing 'Field *' set to 'Request Type', 'Condition *' set to 'equals', and 'Value' set to 'ABR Request for Access'. There are 'Back' and 'Next' buttons at the bottom.

3. Action 1 - set fields:

The screenshot shows a rule configuration interface. On the left, a vertical flow of components is shown: 'When: Issue created', 'Request Type equals', 'Then: Edit issue fields' (highlighted with a blue border), 'And: Transition the issue to WAITING FOR APPROVAL', and 'And: Send email'. On the right, the 'Edit issue' configuration panel is open, showing a 'Choose fields to set...' dropdown menu. Below it, the 'Description' field contains a URL and instructions. The 'Reporter' field is set to '{{issue.user}}'. The 'Approvers' field has 'Jo Approver' selected. There are 'Back' and 'Next' buttons at the bottom.

4. Action 2 - Set Status to **Waiting for Approval** (possibly optional, depending on workflow):

The screenshot shows a workflow editor with five steps:

- When: Issue created** (Rule is run when an issue is created.)
- Request Type equals** (ABR Request for Access)
- Then: Edit issue fields** (Description, Reporter, Approvers)
- And: Transition the issue to** (WAITING FOR APPROVAL) - This step is highlighted with a blue border.
- And: Send email** (myemailaddress@mycompany.com, Issue {{issue.key}} just created and waiting for approval)

The right-hand panel shows the configuration for the 'Transition issue' action:

- Transition issue** (with copy and delete icons)
- Transitions an issue from one status to another, through a workflow. [Learn more about Transition issue action](#)
- Choose the status to transition the issue to:
- Destination status: **WAITING FOR APPROVAL** (dropdown menu)
- Ensure a transition from the issue's source status to your selected destination status exists; [more info](#).
- + add regex to distinguish between multiple transitions to the same status
- Choose fields to set... (dropdown menu)
- More options (expandable)
- Buttons: Back, Next

5. Action 3 - Send email:

The screenshot shows a workflow editor with five steps:

- When: Issue created** (Rule is run when an issue is created.)
- Request Type equals** (ABR Request for Access)
- Then: Edit issue fields** (Description, Reporter, Approvers)
- And: Transition the issue to** (WAITING FOR APPROVAL)
- And: Send email** (myemailaddress@mycompany.com, Issue {{issue.key}} just created and waiting for approval) - This step is highlighted with a blue border.

The right-hand panel shows the configuration for the 'Send email' action:

- Send email** (with copy and delete icons)
- [Learn more about Send email action](#)
- To* (myemailaddress@mycompany.com x dropdown menu)
- Cc Bcc
- Subject* (Issue {{issue.key}} just created and waiting for approval)
- Content* (Issue: {{issue.description}}
Reason: {{issue.reason}}
Requestor: {{issue.reporter}})
- More options (expandable)
- Buttons: Back, Next

Example 2 - Approving or Denying ABR Requests in Jira

ABR-Outgoing-Approval-Accepted-or-Declined ENABLED ✓ Audit log Rule details Update Return to rules

The screenshot shows a Jira Automation rule configuration. On the left, a workflow diagram starts with a trigger 'When: Approval completed' (Rule is run when an approval is accepted or declined). This leads to an 'IF' condition 'If: matches' with the expression `{{approval.decision}} equals Approved`. The 'Then' section contains three actions: 'Send web request' (PUT `https://dc1api.adminbyrequest.com/requests/{{issue.field.customfield_10064}}`), 'Add comment to issue' (ABR Request: `{{issue.fields.customfield_10064}}` Value of Approval: `{{approval.decision}}`), and 'Transition the issue to' (RESOLVED). A '+ Add component' button is at the bottom of the diagram.

Rule details

- Name: ABR-Outgoing-Approval-Accepted-or-Declined
- Description: When a request approval is accepted or declined ...
- Scope: Single project
- Projects: ABR Integration with Jira (ABRJIRA)
- Owner: Steve Dodson
- Actor: Automation for Jira
- Notify on error: E-mail rule owner once when rule starts faili...
- Who can edit this rule?: All admins

1. Trigger:

The close-up shows the 'When: Approval completed' trigger configuration. The trigger is titled 'Approval completed' and has a description: 'For Jira Service Management only. Rule is run when an approval is accepted or declined. This trigger needs no configuration.' Below the description are 'Back' and 'Next' buttons.

2. IF Approved:

When: Approval completed
Rule is run when an approval is accepted or declined

IF

If matches
{{approval.decision}} equals Approved

Then: Send web request
PUT
https://dc1api.adminbyrequest.com/requests/{{issue.fields.customfield_10064}}

And: Add comment to issue
ABR Request: {{issue.fields.customfield_10064}} Value of Approval: {{approval.decision}}

If block

The if block executes the actions within that block when the all specified conditions matches. Otherwise, the following else blocks will be evaluated. [Learn more about If / else block condition](#)

Run actions if...

All conditions match

At least one condition matches

Conditions

> {{approval.decision}} equals Approved

AND

+ Add conditions...

Back Next

3. Action 1 - Send web request (PUT). Important - make sure you read up on API calls [here](#) (API Overview) and [here](#) (Requests API):

When: Approval completed
Rule is run when an approval is accepted or declined

If: matches
{{approval.decision}} equals Approved

Then: Send web request
PUT
https://dc1api.adminbyrequest.com/requests/{{issue.fields.customfield_10064}}

And: Add comment to issue
ABR Request: {{issue.fields.customfield_10064}} Value of Approval: {{approval.decision}}

And: Transition the issue to
RESOLVED

Send web request

This action will send a HTTP request to the url specified. [Learn more](#)

Web request URL *
https://dc1api.adminbyrequest.com/requests/{{issue...

Request parameters must be url encoded, smart values should use: {{value.urlEncode}}.

HTTP method *
PUT

Web request body *
Empty

Delay execution of subsequent rule actions until we've received a response for this web request

Headers (optional)

Key	Value	Hidden
apikey	c32fcbfd-3144-43	<input type="checkbox"/>
Content-length	0	<input type="checkbox"/>

+ Add another header

> Validate your web request configuration

Back Next

> How do I access web request response values in subsequent rule actions?

- Action 2 - Add Comment to Jira issue (Important - make sure you identify the correct customfield ids - yours might be different from the example):

The screenshot shows a workflow rule configuration interface. On the left, a vertical flow of steps is shown:

- When: Approval completed** (Rule is run when an approval is accepted or declined)
- IF** connector
- If: matches** ({{approval.decision}} equals Approved)
- Then: Send web request** (PUT https://dc1api.adminbyrequest.com/requests/{{issue.fields.customfield_10064}})
- And: Add comment to issue** (ABR Request: {{issue.fields.customfield_10064}} Value of Approval: {{approval.decision}})

The 'And: Add comment to issue' step is highlighted with a blue border. On the right, the configuration panel for this action is shown:

- Comment on issue** (with copy and delete icons)
- Learn more about Comment on issue action
- Please enter the comment to add:
- Comment*** (text area containing: ABR Request: {{issue.fields.customfield_10064}} Value of Approval: {{approval.decision}})
- Prevent duplicates by only adding this comment once to a particular issue.
- Comment Visibility (dropdown arrow)
- Back and Next buttons

- Action 3 - Set Status to **Resolved**:

The screenshot shows a workflow rule configuration interface. On the left, a vertical flow of steps is shown:

- When: Approval completed** (Rule is run when an approval is accepted or declined)
- IF** connector
- If: matches** ({{approval.decision}} equals Approved)
- Then: Send web request** (PUT https://dc1api.adminbyrequest.com/requests/{{issue.fields.customfield_10064}})
- And: Add comment to issue** (ABR Request: {{issue.fields.customfield_10064}} Value of Approval: {{approval.decision}})
- And: Transition the issue to RESOLVED** (highlighted with a blue border)

The 'And: Transition the issue to RESOLVED' step is highlighted with a blue border. On the right, the configuration panel for this action is shown:

- Transition issue** (with copy and delete icons)
- Transitions an issue from one status to another, through a workflow. Learn more about Transition issue action
- Choose the status to transition the issue to:
- Destination status** (dropdown menu showing RESOLVED)
- Ensure a transition from the issue's source status to your selected destination status exists; more info.
- + add regex to distinguish between multiple transitions to the same status
- Choose fields to set... (dropdown arrow)
- More options (dropdown arrow)
- Back and Next buttons

6. IF Declined:

ELSE IF

Else-if: matches
 {{approval.decision}} equals Declined

Then: Send web request
 DELETE
 https://dc1api.adminbyrequest.com/requests/{{issue.fields.customfield_10064}}

And: Add comment to issue
 ABR Request: {{issue.fields.customfield_10064}} Value of Approval: {{approval.decision}} Reason: {{issue.fields.customfield_10065}}

And: Transition the issue to
RESOLVED

+ Add component

Else block

The else block executes the actions within the block if the preceding if/else-if blocks failed to match their conditions. The else block can contain its own conditions, making it an else-if, or it can be left blank if it is the last else block. [Learn more about If / else block condition](#)

Run actions if...

All conditions match

At least one condition matches

Conditions

> {{approval.decision}} equals Declined

AND

+ Add conditions...

Back
Next

7. Action 1 - Send web request (DELETE). Important - make sure you read up on API calls [here \(API Overview\)](#) and [here \(Requests API\)](#):

The screenshot displays a workflow configuration interface. On the left, a vertical flow of actions is shown under an 'ELSE IF' condition. The first action is 'Else-if: matches' with the condition `{{approval.decision}} equals Declined`. The second action, 'Then: Send web request', is highlighted with a blue border and contains the following details:

- Method: DELETE
- URL: `https://dc1api.adminbyrequest.com/requests/{{issue.fields.customfield_10064}}`

 The subsequent actions are 'And: Add comment to issue' (with a detailed comment template) and 'And: Transition the issue to RESOLVED'. Below these are buttons for '+ Add component' and '+ Add else'. At the bottom left is another '+ Add component' button.

On the right, the configuration panel for the 'Send web request' action is shown:

- Web request URL*:** `https://dc1api.adminbyrequest.com/requests/{{issue.fi`
- Request parameters:** Must be url encoded, smart values should use: `{{value.urlEncode}}`.
- HTTP method*:** Dropdown menu set to DELETE.
- Web request body*:** Dropdown menu set to Empty.
- Delay execution of subsequent rule actions until we've received a response for this web request
- Headers (optional):** A table with columns for Key, Value, and Hidden. One header is defined: Key: `apikey`, Value: `c32fcbfd-3144-43e`, Hidden: . A '+ Add another header' button is below.
- Validation:** A link to 'Validate your web request configuration'.
- Navigation:** 'Back' and 'Next' buttons.
- Help:** A link to 'How do I access web request response values in subsequent rule actions?'.

8. Action 2 - Add Comment to Jira issue (Important - make sure you identify the correct customfield ids as described in "Overview" on page 1 - even if your field names are the same as the example, your customfield ids might be different):

The screenshot shows a workflow configuration for an 'ELSE IF' condition. The condition is 'Else-if: matches' with the expression '={{approval.decision}} equals Declined'. The 'Then' action is 'Send web request' (DELETE) with the URL 'https://dc1api.adminbyrequest.com/requests/{{issue.fields.customfield_10064}}'. The 'And' actions are 'Add comment to issue' and 'Transition the issue to RESOLVED'. The 'Add comment to issue' action is highlighted with a blue border. The configuration for this action is shown in a detail panel on the right:

- Action:** Comment on issue
- Learn more about Comment on issue action**
- Please enter the comment to add:**
- Comment***
 - ABR Request: {{issue.fields.customfield_10064}}
 - Value of Approval: {{approval.decision}}
 - Reason: {{issue.fields.customfield_10065}}
- Prevent duplicates by only adding this comment once to a particular issue.
- Comment Visibility**
- Buttons:** Back, Next

9. Action 3 - Set Status to **Resolved**:

The screenshot shows a workflow configuration for an 'ELSE IF' condition. The condition is 'Else-if: matches' with the expression '={{approval.decision}} equals Declined'. The 'Then' action is 'Send web request' (DELETE) with the URL 'https://dc1api.adminbyrequest.com/requests/{{issue.fields.customfield_10064}}'. The 'And' actions are 'Add comment to issue' and 'Transition the issue to RESOLVED'. The 'Transition the issue to RESOLVED' action is highlighted with a blue border. The configuration for this action is shown in a detail panel on the right:

- Action:** Transition issue
- Transitions an issue from one status to another, through a workflow. Learn more about Transition issue action**
- Choose the status to transition the issue to:**
- Destination status:** RESOLVED
- Ensure a transition from the issue's source status to your selected destination status exists; [more info](#).
- + add regex to distinguish between multiple transitions to the same status
- Choose fields to set...**
- More options**
- Buttons:** Back, Next

Document History

Version	Author	Changes
1.0 28 January 2024	Steve Dodson	Initial document release.
1.1 20 March 2024	Steve Dodson	Corrected typos and updated screenshots. Reorganized "Creating Jira Automation Rules" section.
2.0 17 April 2025	Steve Dodson	Updated manual structure and layout. Updated portal menu selections.

Index

A

APPROVED	
Request	16
Approving or Denying ABR Requests in Jira	
Example	21

C

Create a custom Request Type	
Task	8
Create an API key	
Task	2
Create custom fields	
Task	5
Creating Jira automation rules	17

D

DENIED	
Request	16
Description field	11

I

Integration Tasks	5
-------------------------	---

P

Populating issue fields after issue creation	
Example	18
Prerequisites	2

R

Request form tab	11
REST client	3

S

Service request	9
Service request with approvals	9
Setup Jira integration	
Task	12

T

Test the API key	
Task	2
Thunder client	3

U

Updating Jira issues	16
Using the Jira Audit log	17

V

VS Code	3
---------------	---