

Power Markdown for Jira Documentation

Overview	5
Getting Started	6
Navigation	7
Main Menu	7
Admin Menu	8
User roles	8
Variables API	8
Markdown Template Administration	9
Overview	9
User Access	9
Managing Templates	9
Creating and Editing Templates	10
Detail	12
Template Usage	12
User Input	13
Pages	13
Fields	14
Field Default Values	15
Field Types	15
Template Issue Fields	16
Template Editor	17
Using the Template Editor	17
Block Types	19
Markdown Syntax	20
Extended Markdown Syntax	22
Expression Autocomplete	23
Expression Context	23
Execution Steps	24
Snapshots	26
Template Debug	27
Expressions	28
Expression Context	28
System Properties	28
Common Expression Tasks	29
Operators	31
Note about in	32
Ternary operator	33
Native Types	33
Groups	33
Identifiers	33
Arrays	34
Functions	35

String Functions	35
Array Functions	38
Object Functions	39
Date Functions	39
Global Functions	39
Date Format Specifiers	39
Expression Limitations	41
Template Limitations	41
User Input Limitations	41
Template Issue Field Administration	42
Managing Template Issue Fields	42
Creating and Updating Template Issue Fields	42
Template Issue Field Limitations	44
Variables Administration	45
Overview	45
Managing Variables	45
Variable Data Types	45
Creating and Editing Variables	46
Managing Variables via the API	47
Authentication	47
Status Codes	47
Error Responses	47
Create Variable	48
Update Variable	49
Delete Variable	49
Get Variable	50
Get All Variables	50
Variable Limitations	50
Using Markdown Templates	52
Applying to an Issue	52
Dynamic Issue Markdown	60
Deleting Dynamic Markdown	64
Setting Default Dynamic Markdown	64
Overriding Dynamic Markdown Target Settings	65
Dynamic Markdown Limitations	65
Administration	66
User Roles	66
Assigning a Default Role	66
Assigning Roles to a User	66
Changing a Users Roles	67
Deleting User Roles	67
Creating a Custom Role	68
Updating a Custom Role	68
Custom Role Limitations	68

Variables API	68
API Settings	69
Regenerating the API Key	69

Overview

Power Markdown for Jira enables the creation of reusable, data driven markdown templates that can be used to generate static or on demand content for Jira issues.

Templates can be used to generate:

- Content for issue fields such as the issue description or other custom fields which use the wiki renderer.
- Content for new issue comments.
- On demand content displayed in custom panels on the issue screen.

Power Markdown for Jira has the following key features:

- Generate data driven content for issues from previously created markdown templates.
- Drag and drop builder enables the construction of markdown templates using a combination of static markdown text, dynamic data (variables, issue fields and user based input), loops (to iterate over dynamic data) and conditions (if, else if, else) to conditionally generate markdown).
- Expressions enable:
 - Merging data into static markdown and creating markdown syntax from data.
 - Building conditional content (using if, else if, else and loop blocks).
 - Conditionally displaying templates for users to select (based on issue data).
 - Conditionally request input from users to use within the markdown.
- Expression context data (data to use via expressions) is available from various sources:
 - Variables (managed in the user interface or via an API).
 - Issue data (both Jira system fields and custom fields).
 - User input data (form data collected from users).
 - Properties set using the 'Set property' template block (set during template execution).
- Enabling control over which users can manage templates and variables and which users can see templates using roles.
- Template snapshots – the last 10 template saves are stored and can be reloaded and reinstated.

Getting Started

Thank you for using Power Markdown for Jira!

To get you up and running as quickly as possible, we have provided this quick start page.

Power Markdown for Jira enables users to create content for issues based on dynamically generated markdown.

Read an [overview](#) of the application and available features.

Content is generated from templates the user or an administrative user creates. Learn how to [create templates](#).

Templates can be as straightforward as a single block of markdown text or can use many advanced features:

- [Collect user input](#) to use in your template.
- [Configure issue fields](#) to use in your templates.
- [Create variables](#) to use in your templates.
- [Manage variables](#) using an API.
- Learn about [expressions](#) and how to use them to transform and inject data into your templates.

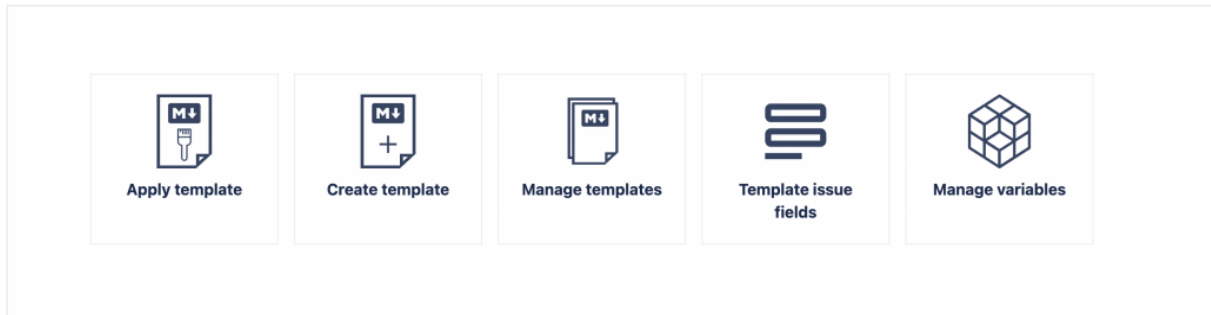
Once a template has been created, it can be used to generate content:

- Use a template to [create a comment or update an issue field](#).
- Use a template to generate and [display content on demand](#).

Navigation

Main Menu

The Power Markdown main menu can be accessed by clicking 'Apps' and then 'Power Markdown'



The menu will change based on a user's role. For example a template administrator will see links to manage and create templates whereas a regular user will only see a link to apply a template.

Each menu item is described below:

Apply template

The apply template link will open a modal window, allowing the user to apply a template to an issue in order to generate content for a specific field (such as the issue description) or to create a new comment for the issue (see [Applying to an Issue](#)).

Create template

Directs the template administrator to the template screen to create a new template (see [Creating and Editing Templates](#)).

Manage templates

Directs the template administrator to the manage templates screen which displays a list of all templates and enables the user to create or manage templates (edit, delete, copy etc.). (see [Managing Templates](#))

Template issue fields

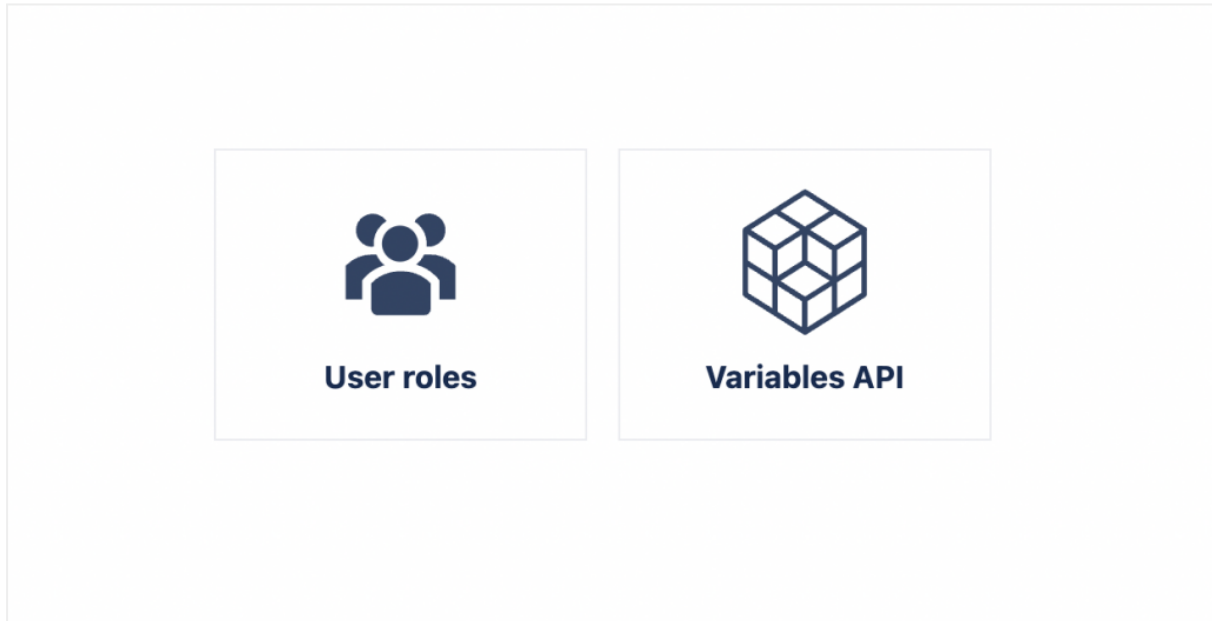
Directs the template administrator to a screen where they can manage template issue fields (these are issue fields which can be used within expressions related to templates). (see [Template Issue Field Administration](#))

Manage variables

Directs the variable administrator to the manage variables screen which displays a list of all variables and enables the user to create or manage variables (edit, delete, copy etc.). (see [Variables Administration](#))

Admin Menu

The admin menu can be accessed by clicking 'Apps', 'Manage your apps' and then clicking 'Power Markdown' under the apps section in the left hand menu.



User roles

Directs the user to the user roles screen, which enables users to manage roles, assigning roles to users and setting a default role for all users. See [User Roles](#).

Variables API

Directs the user to the variables API screen, which enables users to retrieve the variables API key (for access to manage variables via an API). See [Variables API](#).

Markdown Template Administration

Overview

Markdown templates enable users to create reusable, data driven content for issues.

Templates can be used to generate:

- Content for issue fields such as the issue description or other custom fields which use the wiki renderer.
- Content for new issue comments.
- On demand content displayed in panels on the issue screen.

User Access

Users with the template administrator role can manage templates. Jira administrators can set up custom roles or use the built in system roles to assign to individual users to control access.

Administrators can also set a default role for all users.

See [User Roles](#).

Managing Templates

To manage templates, click on the 'Manage Templates' link on the Power Markdown main menu.

The screenshot shows the 'Templates' management interface. At the top, there are two buttons: 'Back to menu' (callout 2) and 'Create template' (callout 1). Below these is a filter section with a dropdown menu labeled 'Name' (callout 3) and a text input field labeled 'Filter' (callout 4). The main area is a table with columns: 'Name', 'Description', 'Visible to roles', and 'Created'. The 'Created' column has a dropdown arrow (callout 5). The first row of the table has 'Template Name' in the 'Name' column (callout 6), 'Template Description' in the 'Description' column, 'Custom Role Administrator User' in the 'Visible to roles' column, and '09/10/2022' in the 'Created' column. To the right of the first row is a three-dot menu (callout 7) with 'Copy' (callout 7) and 'Delete' (callout 8) options. At the bottom left, there is a pagination control with a left arrow (callout 9), a box containing the number '1', and a right arrow.

1. Create template – click to navigate to the template screen to create a template.
2. Back to menu – click to go back to the main menu.
3. Filter field selection – choose the field you wish to filter on.
4. Filter field value – enter a value to automatically filter the grid based on the selected filter field.

5. Sort grid – click on the name, description or created on column headers to sort the rows.
6. Edit template – click on the template name to navigate to the template edit screen.
7. Copy template – prompts for a template name and copies the chosen template.
8. Delete template – confirms you wish to delete the template and after confirmation deletes the chosen template.
9. Page templates – click on the page number to navigate to the chosen page.

Creating and Editing Templates

Navigate to the template screen by either 1. clicking the 'Create Template' button on the Power Markdown main menu or 2. by clicking on the 'Manage Templates' link on the Power Markdown main menu and then the 'Create template button' above the templates grid (to create) or on an existing template's name (to edit).

Creating and editing a template follows an identical process and are explained in brief below (the rest of this section explains each of the steps in more detail).

Enter a name (1) (required) and description (2) (optional) for the template.

Select the roles (3) the template is visible to (required).

Choose any variables (4) you wish to use in the template and once complete click 'Rebuild variable collection' (5) to fetch the variables and add them to the expression context.

Note: You can amend the variables you wish to use in the template at any time by adding or removing and clicking 'Rebuild variable collection'.

The screenshot shows the 'Unnamed template' form in Jira. It features a top navigation bar with 'Save', 'Save and close', and 'Close' buttons. Below this is a tabbed interface with 'Detail', 'Usage', 'User input', 'Issue fields', and 'Template' tabs. The 'Detail' tab is selected. The form contains the following fields and controls:

- Name *** (1): A text input field with a red circle containing the number 1.
- Description** (2): A text input field with a red circle containing the number 2.
- Visible to roles *** (3): A dropdown menu with 'Select...' and a red circle containing the number 3.
- Variables** (4): A dropdown menu with 'Select...' and a red circle containing the number 4.
- Rebuild variable collection** (5): A button with a red circle containing the number 5.

Specify how the template can be used in Jira (optional, see [Template Usage](#)). This controls which issues the template can be used on – the display condition (1), whether it can be used to set fields, create comments or both (2) and whether the end user can edit the generated content before applying (3).

Save Save and close Close Unnamed template

Detail Usage User input Issue fields Template

The settings below control how the template can be used when generating content for issue fields or comments.

Display Condition 1

Apply template target 2
Field or Comment

Apply template preview 3
Edit

Setup any user input (optional, see [User Input](#)).

Save Save and close Close Unnamed template

Detail Usage User input Issue fields Template

Create page

1 No pages have been created. Click 'Create page' above to create a page.

Add any template issue fields (optional, see [Template Issue Fields](#)).

Save Save and close Close Unnamed template

Detail Usage User input Issue fields Template

⚠ Note: The template issue fields configured below are shared between all templates and changes here will affect all templates which use these fields.

Create template issue field

Field name	Context property	Default value	
Priority	priority	Default priority	...
Summary	summary	Default summary	...

Build the template (see [Template Editor](#)).

Save Save and close Close Unnamed template

Detail Usage User input Issue fields Template

Editor Expression context Execution steps

Markdown New line If Else if Else Loop Exit Set context property

Click 'Save' or 'Save and close'.

Detail

The detail tab enables the user to provide descriptive information about the template (a name (1) and description (2), to select which user roles the template is visible to (3) and to select which variables are available to use in the template expressions (4).

The screenshot shows the 'Detail' tab of a template configuration interface. At the top, there are buttons for 'Save', 'Save and close', and 'Close', and the title 'Unnamed template'. Below the tabs, there are four main sections: 'Name' with a text input field (marked with a red circle 1), 'Description' with a text input field (marked with a red circle 2), 'Visible to roles' with a dropdown menu (marked with a red circle 3), and 'Variables' with a dropdown menu (marked with a red circle 4). At the bottom, there is a 'Rebuild variable collection' button (marked with a red circle 5).

Note: After adding or removing variables it is required to click on the 'Rebuild variable collection' button.

Template Usage

The template usage tab provides options related to using the template.

The screenshot shows the 'Usage' tab of a template configuration interface. At the top, there are tabs for 'Detail', 'Usage', 'User input', 'Issue fields', and 'Template'. Below the tabs, there is a heading: 'The settings below control how the template can be used when generating content for issue fields or comments.' There are three main sections: 'Display Condition' with a text input field (marked with a red circle 1), 'Apply template target' with a dropdown menu showing 'Comment' (marked with a red circle 2), and 'Apply template preview' with a dropdown menu showing 'Edit' (marked with a red circle 3).

Display condition (1)

The display condition is an expression which when set will filter the available templates based on the expression result (false, an empty string, 0, undefined or null equate to a false value and all other values equate to true).

Note: The expression context for the display condition is built from issue data only and variables are not available to use.

This enables the template administrator to restrict access to the template based on any of the created template issue fields. Such as only displaying the template for selection if the selected issue has a specific priority.

See [Expressions](#).

Apply Template Target (2)

The apply template target enables the template administrator to control whether the template can be used to update an issue field, create a new comment or both.

Apply Template Preview (3)

The apply template preview, controls whether the generated content can be edited by the user prior to being used to update a field or create a comment or whether it is displayed as a read-only preview.

User Input

The user input tab enables the template administrator to define one or more pages of user input (enabling capture of data from the template end user), which is added to the expression context to be used in the templates.

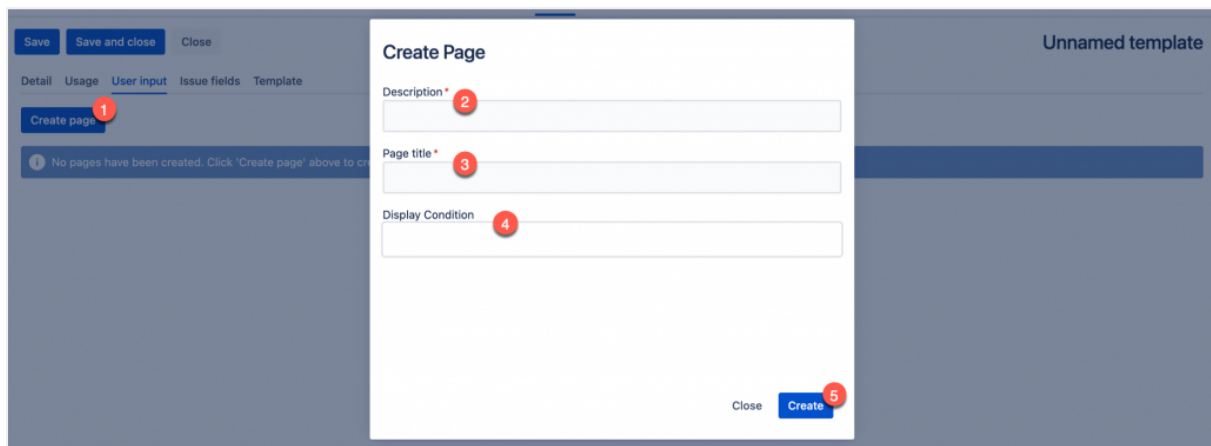


Pages

Note: Each page can optionally use a display condition to determine whether it should be visible to the user (the expression context includes the current target, issue data (configured template issue fields), selected variables as well as any user input data from previous pages).

To create/edit a user input page follow the steps below:

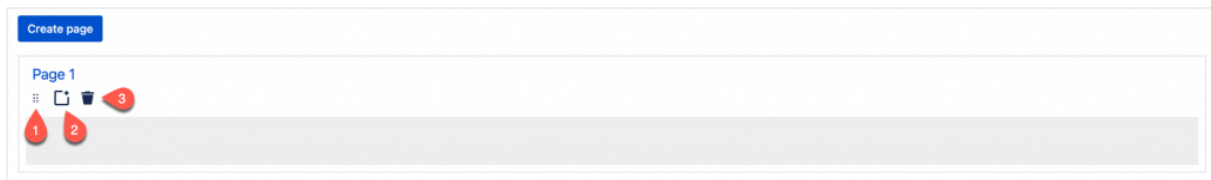
1. Click on the 'Create page' button.
2. Enter a description (required)
3. Enter a page title (required).
4. Enter a display condition if required (optional).
5. Click 'Create' (if creating) or 'Update' (if editing).



Pages can be re-ordered by clicking and dragging on the drag handle below the page title (1).

Pages can be deleted by clicking on the bin icon under the page title (3).

To add fields to the page, click on the 'Create field' icon next to the drag handle (2).

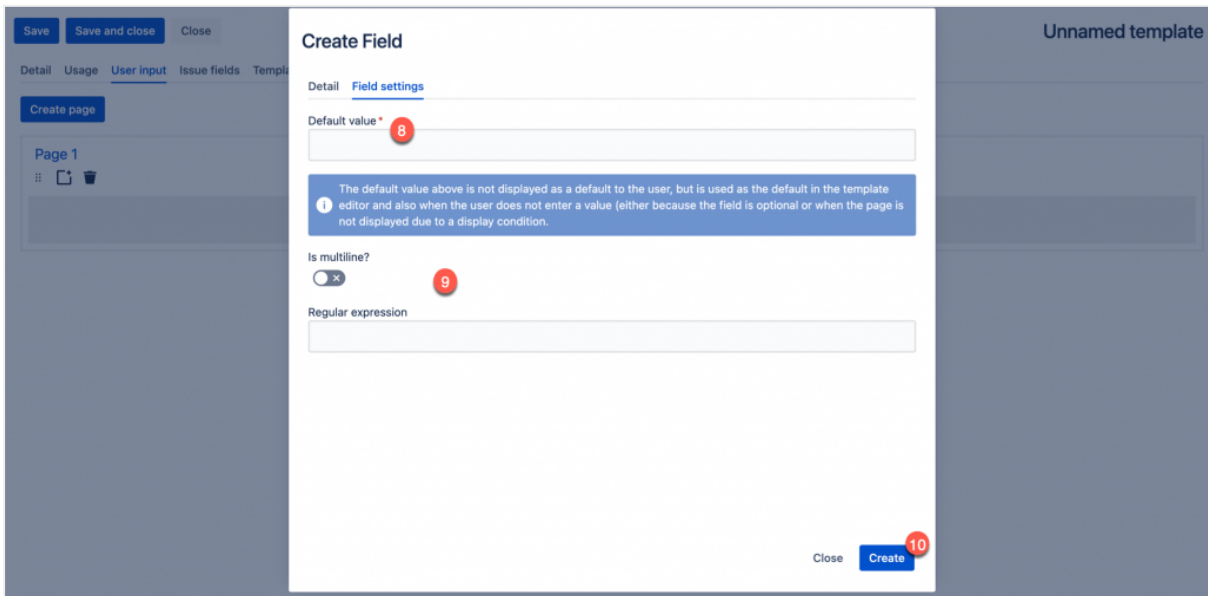
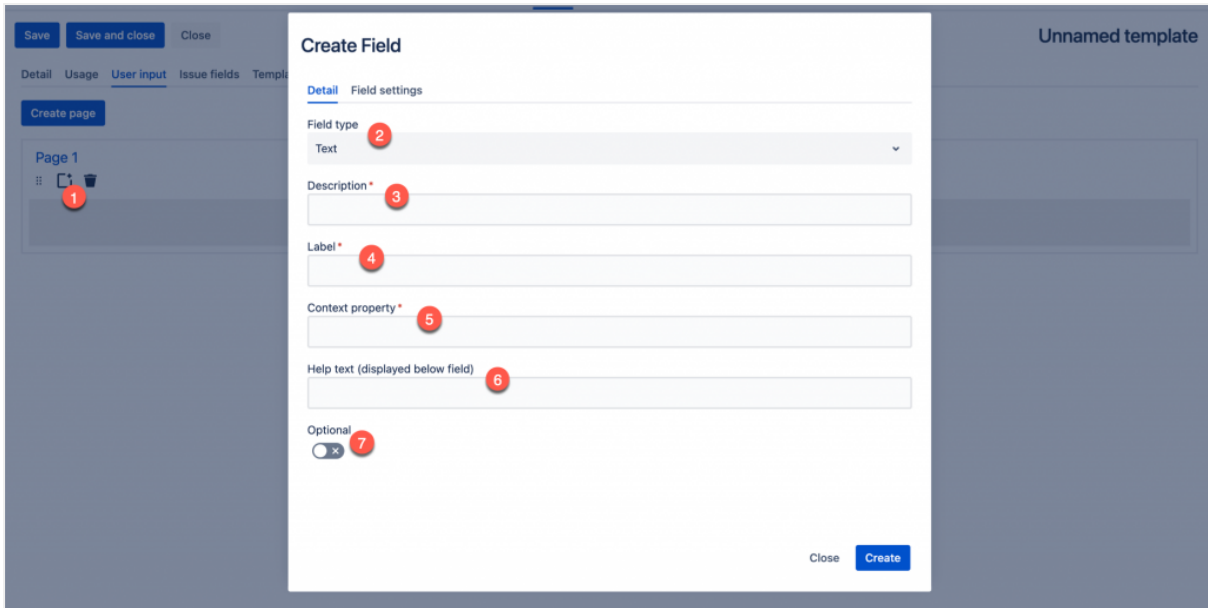


Fields

Fields enable the capture of specific user data.

The process for creating and editing fields is identical.

1. Click on the 'Create field' icon on the page you wish to add a field for.
2. Select the field type (see field types below for more information) (required).
3. Enter a description (required).
4. Enter a label (required) – this is displayed above the field.
5. Enter the context property name. This is the name that is used to access the data in the expression context.
6. Enter the help text (displayed under the field) (optional).
7. Determine whether the field is optional.
8. Enter a default value for the field based on the type (see field types below).
9. Configure any other required or optional settings based on the field type (see field types below).
10. Click 'Create' or 'Update'.



Field Default Values

Each field requires a default value. This value is not displayed as a default to the user, but is used as the default in the template editor expression context and also in the template expression context when the user does not enter a value (either because the field is optional or when the page is not displayed due to a display condition).

Note: Default values are not subject to validation settings detailed below.

Field Types

There are seven types of data which can be collected from users. These fields have type specific settings which are detailed below:

Text

Settings	Description
----------	-------------

Is Multiline	Switches between a single line input to a multi line input.
Regular expression	Validates the input to ensure it matches the expression entered.

Number

Settings	Description
Min value	The minimum numeric value which is accepted.
Max value	The maximum numeric value which is accepted.

Boolean

No additional settings available.

Date Time

Settings	Description
Min value	The minimum date time value which is accepted.
Max value	The maximum date time value which is accepted.

Date

Settings	Description
Min value	The minimum date value which is accepted.
Max value	The maximum date value which is accepted.

Single Select and Multiple Select

Both single and multiple select are similar so are detailed together, with the difference that multiple select allows for multiple entries for a default.

Settings	Description
Select options	The select options can be entered manually (one per line) or generated at runtime using an expression which must evaluate to an array of strings (note the array mapProperty and object getPropertyNames functions can help to retrieve object property names if required. See Expressions).

Template Issue Fields

Template Issue fields enable the template administrator to define issue data they wish to use in the templates (via the expression context).

Template issue fields are not linked to any specific templates and can be created via the Power Markdown main menu, but are included in the template editor for convenience.

For more information see [Template Issue Field Administration](#).

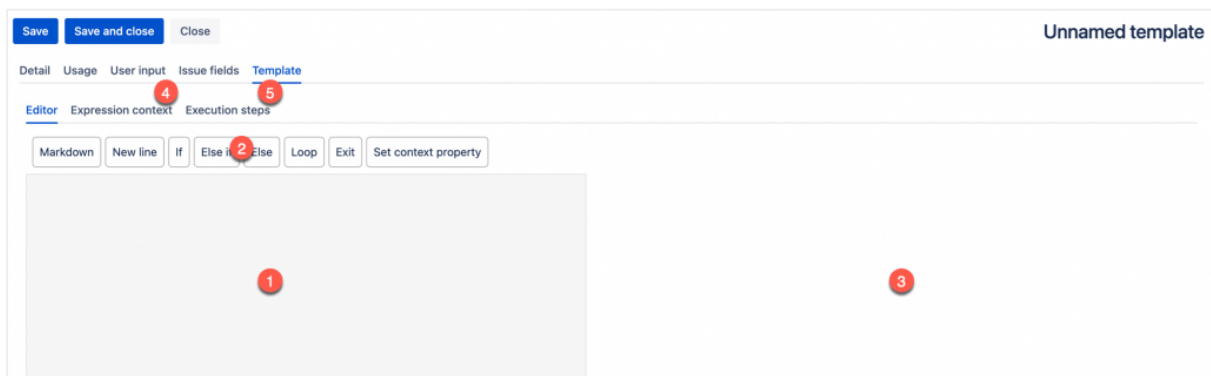
Template Editor

The template editor enables construction of the markdown template using a drag and drop interface.

It is an adjustable split view, with the left hand – the canvas (1) allowing for the construction of the template using template blocks (2) (draggable blocks used to construct the markdown template document) and the right hand side showing a preview (3) of the generated content.

The 'Expression context' tab (4) (see [Template Editor – Expression Context](#)) enables viewing and temporary modifications to the expression context.

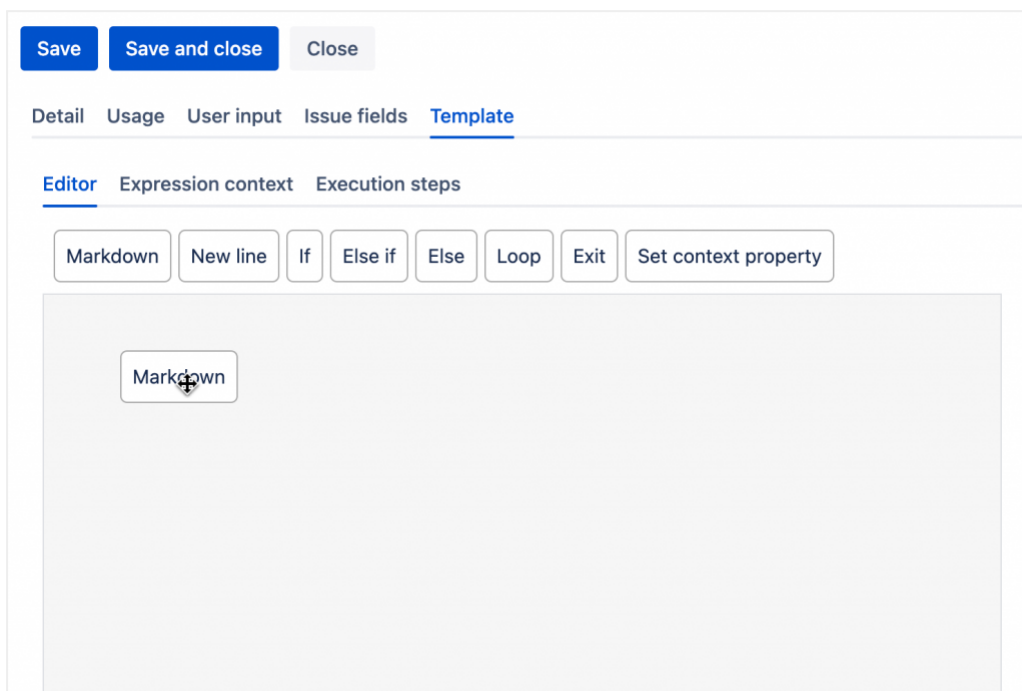
The 'Execution steps' tab (5) (see [Template Editor – Expression Steps](#)) provides a breakdown of the steps taken to generate the markdown based on the blocks added to the canvas.



Using the Template Editor

Adding a Block

A block can be added to the template by dragging it from the block toolbar onto the editor canvas at the desired position.



'Else If' and 'Else' blocks have specific positions they can be added.

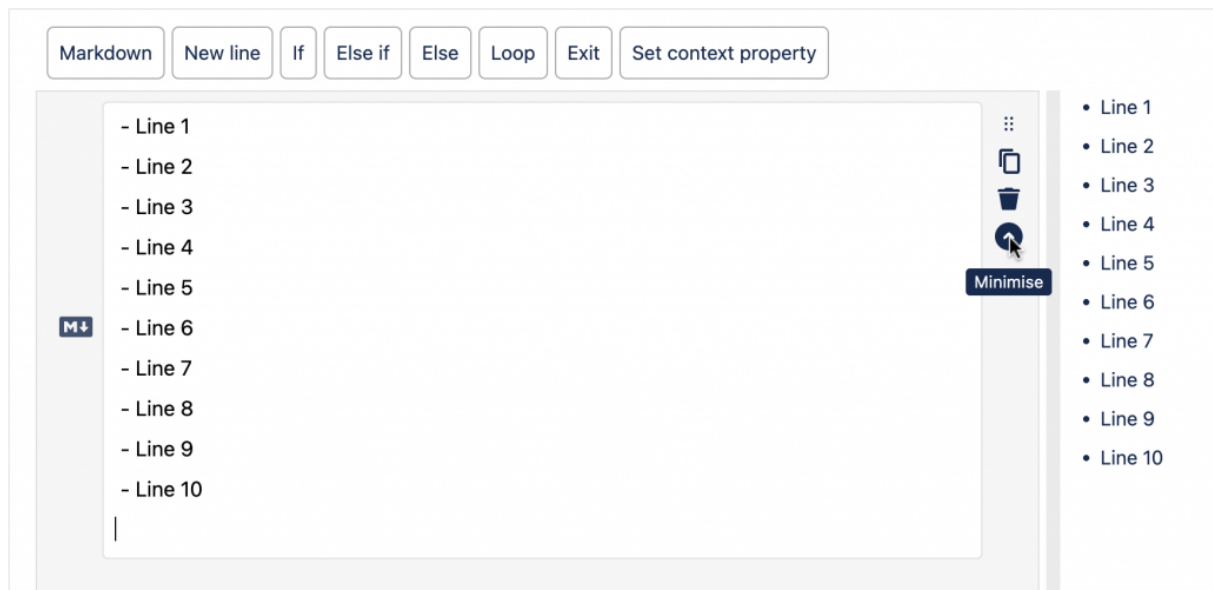
'Else If' blocks can only be added between a condition starting and the corresponding 'end' (if no 'Else' exists), or before the 'Else' if one exists.

'Else' can only be added between 'If' and the corresponding 'end' (if no 'Else If' exists), or after the last 'Else If' and before the corresponding 'end'. An 'Else' cannot be added if an 'Else' already exists for the condition.

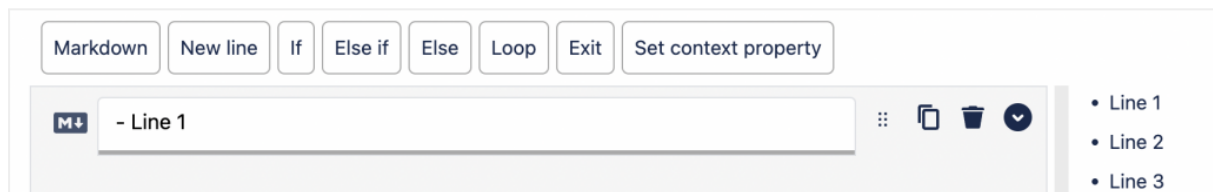
Minimising/Maximising Markdown blocks

Markdown blocks longer than 10 lines can be minimised and maximised by clicking on the up/down arrow icons in the block menu.

Note: Minimising a large Markdown block helps when reordering blocks or when you have a large template and want to hide certain parts of it.

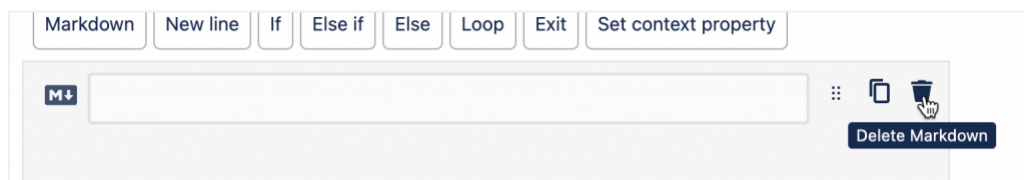


When minimised, the block has a darker bottom border:



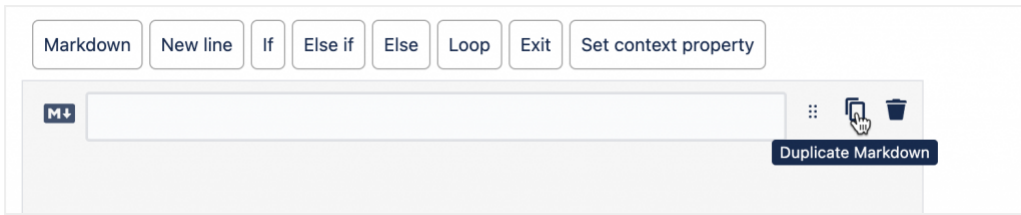
Deleting a Block

A block can be deleted by clicking on the trash can icon to the right of the block when hovering over it.



Duplicating a Block

A block can be duplicated by clicking on the duplicate icon to the right of the block when hovering over it.

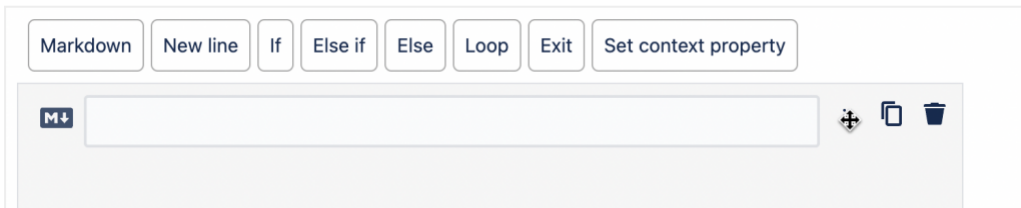


Duplicating an 'If' and 'Else if' block will duplicate the block and all corresponding child blocks.

An 'Else' block cannot be duplicated.

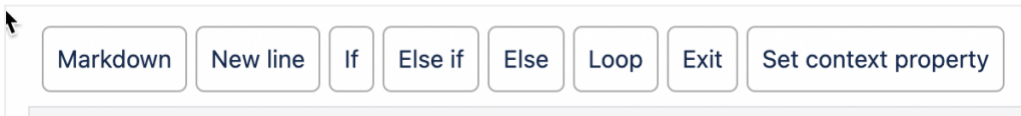
Moving a Block

A block can be moved by clicking on the drag icon to the right of the block and dragging it into the desired position.



'Else if' and 'Else' blocks can only be dragged into valid positions based on the rules defined above.

Block Types



There are eight block types, each of which are described below:

Block	Description
Markdown	<p>Multi line text area allowing for the input of markdown syntax and extended markdown syntax (see Markdown Syntax and Extended Markdown Syntax).</p> <p>As well as markdown, the input also supports expression blocks (see Expressions).</p> <p>Expression blocks start and end with the dollar character '\$'.</p> <p>Note: dollar characters can be escaped by preceding them with a backslash: '\$'.</p> <p>Example markdown and expression block: The issue summary is: **\$issue.summary\$**</p> <p>Would print the text 'The issue summary is:' and then inject the summary template issue field context property into the markdown in bold (see Template Issue Field Administration).</p>
New line	Creates a newline

If	Condition evaluates an expression and executes the child blocks contained within it if it evaluates to a truthy value (false, an empty string, 0, undefined or null equate to a false value and all other values equate to true).
Else If	Condition evaluates an expression and executes the child blocks contained within it if it evaluates to a truthy value (false, an empty string, 0, undefined or null equate to a false value and all other values equate to true).
Else	Condition executes the blocks contained within it if all previous paths have been evaluated to false.
Loop	<p>Executes a loop from a numeric expression to another numeric expression.</p> <p>Within a loop the 'loopIndices' numeric array system property is used to access the current index.</p> <p>Example: 'Loop from' value: 0 'Loop to' value: 9 The Loop block will iterate 10 times from 0 through 9.</p> <p>Within the Loop block the loopIndices context property will contain the loop index at the current nesting level. If there is just one loop then the index will be accessed like so: loopIndices[0]. If there is a loop nested within another loop then the inner loop can be accessed like so: loopIndices[1] etc.</p> <p>Note: The Loop steps enables looping through arrays by providing the start index as the from expression value and the array length as the to expression value (length can be accessed using the length function arrayProperty length. See more about functions in Expressions).</p>
Exit	Terminates the markdown generation at the current point.
Set Context Property	<p>Sets a context property to the expression value.</p> <p>Requires a path and expression value.</p> <p>The path can use dot notation to set a child property as is required.</p> <p>Note: Properties set in If, Else If, Else or Loop blocks are not scoped to those blocks and are available after the block exits.</p>

Type	Markdown
Headings	# Heading 1 ## Heading 2 ### Heading 3 #### Heading 4 ##### Heading 5 ##### Heading 6
Strong/Bold	**strong**
Emphasis/Italics	<i>*emphasis*</i>
Superscript	^{^superscript^}
Subscript	_{~subscript~}
Underline	<u>+under line+</u>
Strikethrough	-strike through-
Blockquote	> I am a block > **I am a bold block**
Horizontal Rule	---
Hyperlink	[Link Text](URL) [**Strong Link**](URL) – can use limited markdown in link text
Image	
Ordered lists	1. First item 2. Second item 3. Third item 1. Indented item 2. Indented item 1. Indented Item 2. Indented Item 3. Indented item 4. Fourth item

Unordered lists	<ul style="list-style-type: none"> * First item * Second item * Third item <ul style="list-style-type: none"> * Indented item * Indented item <ul style="list-style-type: none"> * Indented item * Indented item * Indented item * Fourth item <p>Note: unordered lists can start with -, * or +</p> <p>Note: to start an unordered list with a number – 1\. First Item</p>
Inline code	... `inline code` ...
Code block	<pre>```javascript console.log('Hello world!'); ```</pre>
Url	< https://www.jetpacksoftware.com >
Email Address	< support@jetpacksoftware.com >

Extended Markdown Syntax

Extended markdown syntax enables the creation of content such as coloured text, dates, status', mentions and panels.

Type	Markdown
Coloured text	<p>{c:HEX_COLOUR:TEXT}</p> <p>Example: {c:#ff0000:some text}</p> <p>Note: You can surround with other markdown: <code>***-+{c:#ff0000:coloured text}-+***</code></p>
Date	<p>{d:DATE}</p> <p>Example: {d:2022-01-01}</p> <p>Note: Will attempt to convert any date or time stamp.</p>

Status	<p>{s:TYPE:TEXT}</p> <p>Example: {s:blue>Status text}</p> <p>Types: purple, blue, red, yellow, green and gray (defaults to gray when an invalid type is passed)</p>
Mention	<p>{m:USER_NAME or ACCOUNT_ID} – {m:Adam Hills}</p>
Panel	<p>{p:TYPE}CONTENT{p}</p> <p>Example: {p:info}I am an info panel!Regular, Strong, <i>Italic</i>, <i>Strong Italic</i>, +-other marks+-(p)</p> <p>Types: info, note, success, warning and error.</p>

Expression Autocomplete

The editor expression autocomplete will provide hints based on the defined input, template issue fields and selected variables along with other system properties.

The autocomplete will update to reflect any 'Set Context Property' blocks that have been added.

Within a condition path, if a context property is set then the system will attempt to infer the structure of the value assigned to the path and will provide autocomplete support accordingly.

The autocomplete will always use the most recent value set, unless that value is undefined or null and set in a condition in which case it will take the last defined value.

Expression Context

The expression context tab provides visibility of the expression context and also enables modifications to the context in order to test the template with different values.

Note: Saving is disabled when the expression context has been modified. To enable saving again, the expression context must be reset.

To make changes to the expression context, update the context JSON using the text input (1). The value entered must be valid JSON and an error will be displayed if it is invalid.

After making changes to the JSON, click the 'Update' button (2).

To reset the expression context, click on the 'Reset' button.

Any changes to the variable collection, user input or template issue fields, will reset the context.

Save Save and close Close Unnamed template

Detail Usage User input Issue fields Template

Editor Expression context Execution steps

Update Reset

1 Any changes to the variable collection, user input or issue fields will cause any changes made below to be lost.

```
{
  "issue": {
    "summary": "Default summary",
    "key": "PK-1",
    "projectKey": "PK",
    "projectName": "Project Name"
  },
  "input": {},
  "target": "editor",
  "indices": []
}
```

Execution Steps

The execution steps tab provides a breakdown of the steps taken to generate the markdown based on the blocks added to the canvas.

Jira Software Your work Projects Filters Dashboards People Apps Create

Save Save and close Close Load a save... ↕ ↺

Detail Usage User input Issue fields Template

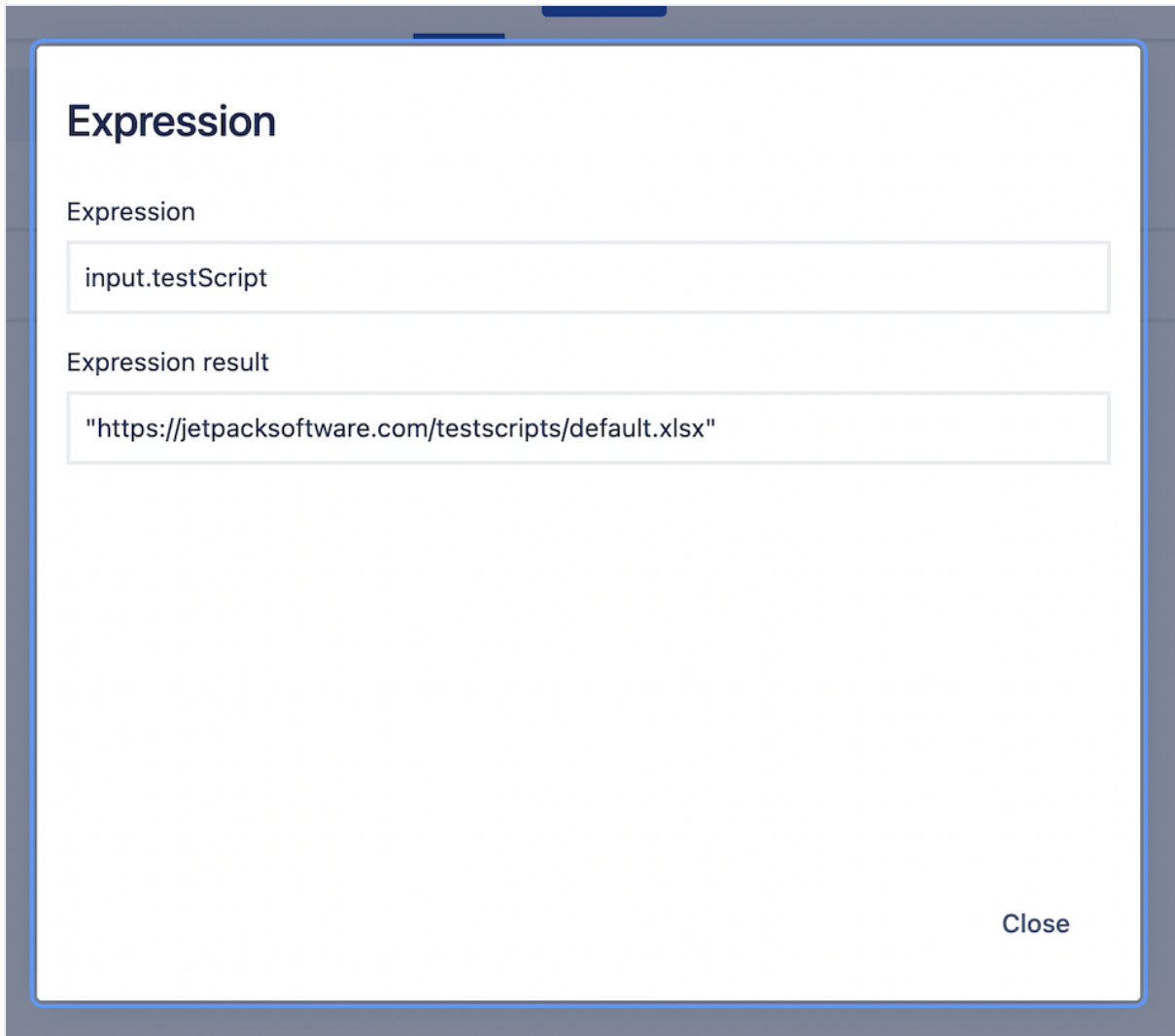
Editor Expression context Execution steps

Markdown	# Feature Test Results
New line	
Markdown	[Test Script](
Expression	input.testScript
Markdown)
If	input.passed
▼ Else	
Markdown	{p:error}### Status: Failed{p}
New line	
Markdown	### Issue Description:
New line	
Expression	input.issueDescription escape ()
New line	
Markdown	### Steps to reproduce:
New line	
Expression	input.stepsToReproduce escape ()

This can be useful to analyse the steps taken to generate the Markdown.

Note: Markdown blocks containing expressions are split into Markdown and expression steps to provide better visibility of the expressions within the markdown.

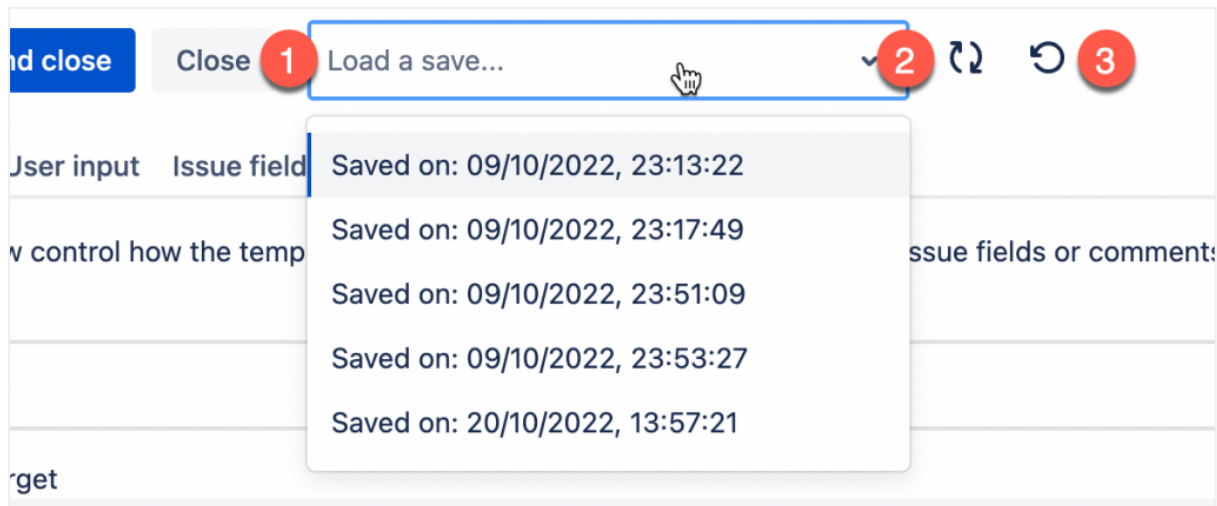
The following blocks: Expression, If, Else If, Loop and Set Context Property all provide additional information available by clicking on the step name.



Snapshots

Each time a template is saved, a snapshot of that template is also saved enabling a user to go back and view the last 10 saves that have been made.

The Snapshot selection dropdown is found at the top of the template editor screen, next to the menu buttons.



To load a snapshot, select one from the drop down and it will automatically load.

Note: If there are unsaved changes to the current template a notification is provided.

After loading a snapshot a user can make any changes required and then save. Saving a snapshot will overwrite the latest template save (a notification is displayed).

After the snapshots have initially loaded, to reload any new saved snapshots, click the refresh button (2).

To reset back to the most recent save, click the reset button (3).

Template Debug

The template debug feature enables users to generate debug data to provide Power Markdown support for debugging purposes on a specific template.

It also provides the ability to debug problems which occur when your own users are attempting to use the templates that have been created.

If a template fails to render (this could be caused by a deleted variable or template issue field), the user will be presented with an error screen which includes encoded template debug data for them to send to the template administrator for review.

To access the template debug feature:

1. Navigate to the template screen (in either create or edit mode).
2. Ensure the screen has focus by clicking anywhere in the template screen.
3. Press Ctrl + D to display the 'Debug data' (1) button.
4. Click on the 'Debug data' button which will open a modal window.
5. If you are providing data to Jetpack software support
 1. Copy the debug data (2) to send over.
 2. Click the 'Close' button.
6. If you are debugging an issue with one of your templates:
 1. Delete the existing debug data and paste in the data provided by one of your users (2).
 2. Click the 'Load' (3) button.
 3. This will load the template and expression context, enabling review of the template and any errors that are preventing the template from loading.

- Once in debug mode, the template is read only and cannot be changed or saved. The page will need to be refreshed to make any changes to that template again.



Expressions

Power Markdown enables the input of user defined expressions to generate dynamic markdown and build conditional logic to control the visibility of templates and other items such as templates and user input pages.

Expression Context

Expressions make use of a data structure 'the expression context'.

The Expression context is an object constructed from user defined variables (see [Variable Administration](#)), issue data (see [Template Issue Field Administration](#)), user input data (see [User Input](#)), and other system defined properties.

Note: User defined variables are located in the root of the expression context for quick access.

System Properties

Name	Description
loopIndices	Provides the current loop iteration index when using 'Loop' blocks. See 'Loop' block .
input	Provides access to input properties, defined using the template input section. See User Input .
issue	Provides access to issue data. The fields available are configured by the template administrator. See Template Issue Fields .

target	<p>Indicates the current template execution target. Possible values:</p> <p>'editor' – when executing within the template editor. 'dynamic-markdown' – when applying a template as dynamic markdown. 'field' – when applying a template to a field. 'comment' – when creating a comment.</p> <p>Note: This enables you to define conditions which check the target and provide variations in the Markdown or user input based on the active target.</p>
--------	--

Common Expression Tasks

Task	Detail
Invoking a function	<p>There are two types of functions which can be invoked in expressions, functions associated with properties and global functions.</p> <p>Global functions can be invoked by calling the function by name. Example:</p> <pre>now()</pre> <p>Property functions can be invoked by using the 'pipe' character after the property name. Example:</p> <pre>stringProperty toUpperCase()</pre> <p>See Functions.</p>
Guarding for undefined context properties	<p>When you wish to check if a context property exists, use the <code>isDefined</code> global function.</p>
Converting an array of objects to a table	<p>Use the <code>toTable</code> array function to convert an array of objects into a table structure, where each object in the array is displayed as a row in the table.</p> <p>This function also enables custom headers for cell headers and custom cell template expressions where you can specify an expression for the cell providing access to the cell's object properties.</p>

<p>Mapping properties from an array of objects</p>	<p>When you have an array of objects and you wish to retrieve a map of specific property values from those objects you can use the mapProperty array function.</p> <p>This is useful when generating select list options for user input and you only wish to retrieve one value from an array of objects.</p>
<p>Working with dates</p>	<p>Date variables and user input dates are stored as timestamps.</p> <p>To format a timestamp as a date string, use the formatDate number function.</p> <p>To format a date string use the formatDate string function.</p> <p>To format the current date use the formatDate global function.</p> <p>To retrieve the current date as a timestamp use the now global function.</p>
<p>Filtering object arrays</p>	<p>Object arrays can be filtered to find specific items. See Arrays for more information on filtering arrays of objects.</p>
<p>Array iteration</p>	<p>To iterate items in an array you can use the loop block (See Block Types).</p> <p>Within the loop block the loopIndices context property will contain the loop index at the current nesting level. If there is just one loop then the index will be accessed like so: loopIndices[0]. If there is a loop nested within another loop then the inner loop can be accessed like so: loopIndices[1] etc.</p> <p>Note: The loop step enables array iteration by providing the start index as the from expression value and the array length as the to expression value (length can be accessed using the length function arrayProperty.length()).</p> <p>Within the body of the loop you can access the current array item like so: arrayProperty[loopIndices[n]] where n is the current loop nesting level.</p> <p>Example:</p>

Editor Expression context Execution steps

Markdown New line If Else if Else Loop Exit Set context property

```

</> array = [1,2,3,4,5]
LOOP FROM: 0 TO: array|length() - 1
</> arrayItem = array[loopIndices[0]]
M+ Array item is: $arrayItem$
↓
END

```

Set context property scope

Properties set in If, Else If, Else or Loop blocks are not scoped to those blocks and are available after the block exits.

Conditionally setting a context property

When conditionally setting a context property where the result of the expression is undefined within the editor, the editor is unable to determine the correct value or type.

Accessing the value and any properties on it (if it is an object) will display validation errors within the editor canvas.

To resolve this, use a condition block which checks to see if the target is the editor (target == 'editor') and set the context property within this condition block to the required value.

Example:

```

IF target == 'editor'
</> client = clients[0]
ELSE
</> client = clients[.name == issue.client][0]
END

```

Operators

Type	Operation	Symbol
Unary		

	Negate	!
Binary		
	Add/Concat	+
	Subtract	-
	Multiply	*
	Divide	/
	Divide and floor	//
	Modulus	%
	Power of	^
	Logical AND	&&
	Logical OR	
Comparison		
	Equal	==
	Not equal	!=
	Greater than	>
	Greater than or equal to	>=
	Less than	<
	Less than or equal to	<=
	Element in array or string	in

Note about in

The in operator can be used to check for a substring: "Cad" in "Ron Cadillac", and it can be used to check for an array element: "coarse" in ['fine', 'medium', 'coarse']. However, the == operator is used behind-the-scenes to search arrays, so it should not be used with arrays of objects. The following expression returns false: {a: 'b'} in [{a: 'b'}].

Ternary operator

Conditional expressions check to see if the first segment evaluates to a truthy value. If so, the consequent segment is evaluated. Otherwise, the alternative is. If the consequent section is missing, the test result itself will be used instead.

Expression	Result
"" ? "Full" : "Empty"	Empty
"foo" in "foobar" ? "Yes" : "No"	Yes
{agent: "Archer"}.agent ? "Kane"	Archer

Native Types

Type	Examples
Booleans	true, false
Strings	"Hello world", 'Hello world'
Numerics	6, -7.2, 5, -3.14159
Objects	{hello: "world!"}
Arrays	['hello', 'world!']

Groups

Parentheses work how you would expect them to:

Expression	Result
(83 + 1) / 2	42
1 < 3 && (4 > 2 2 > 4)	true

Identifiers

Access variables in the context object by typing their name. Objects can be traversed with dot notation, or by using brackets to traverse to a dynamic property name.

Example context:

```
{
  name: {
    first: "Malory",
    last: "Archer"
  },
  exes: [
```

```

    "Nikolai Jakov",
    "Len Trexler",
    "Burt Reynolds"
  ],
  lastEx: 2
}

```

Expression	Result
name.first	Malory
name['la' + 'st']	Archer
exes[2]	Burt Reynolds
exes[lastEx - 1]	Len Trexler

Arrays

Arrays of objects can be filtered by including a filter expression in brackets. Properties of each collection can be referenced by prefixing them with a leading dot. The result will be an array of the objects for which the filter expression resulted in a truthy value.

Example context:

```

{
  employees: [
    {first: 'Sterling', last: 'Archer', age: 36},
    {first: 'Malory', last: 'Archer', age: 75},
    {first: 'Lana', last: 'Kane', age: 33},
    {first: 'Cyril', last: 'Figgis', age: 45},
    {first: 'Cheryl', last: 'Tunt', age: 28}
  ],
  retireAge: 62
}

```

Expression	Result
employees[.first == 'Sterling']	[{first: 'Sterling', last: 'Archer', age: 36}]
employees[.last == 'Tu' + 'nt'].first	Cheryl
employees[.age >= 30 && .age < 40]	[{first: 'Sterling', last: 'Archer', age: 36},{first: 'Lana', last: 'Kane', age: 33}]
employees[.age >= 30 && .age < 40][.age < 35]	[{first: 'Lana', last: 'Kane', age: 33}]

employees[.age >= retireAge].first	Malory
------------------------------------	--------

Functions

String Functions

Name	Description	Parameters	Return Value
toUpperCase	Returns the calling string value converted to uppercase.	N/A	string
toLowerCase	Returns the calling string value converted to lowercase.	N/A	string
toTitleCase	Returns the calling string value converted to title case.	N/A	string
trimStart	Removes whitespace or specified characters from the beginning of the calling string (if no chars passed then defaults to whitespace)	chars: string (string of characters to remove from the beginning of the calling string)	string
trimEnd	Removes whitespace or specified characters from the end of the calling string (if no chars passed then defaults to whitespace)	chars: string (string of characters to remove from the end of the calling string)	string
trim	Removes whitespace or specified characters from the beginning and end of the calling string (if no chars passed then defaults to whitespace)	chars: string (string of characters to remove from the beginning and end of the calling string)	string
startsWith	Determines whether a string begins with the characters of a specified string, returning true or false as appropriate.	searchString: string (string to find at the beginning of the calling string) start: number (The position in the calling string at which to begin searching for searchString. Defaults to 0)	boolean

endsWith	Determines whether a string ends with the characters of a specified string, returning true or false as appropriate.	searchString: string (string to find at the end of the calling string) length: number (If provided, it is used as the length of the calling string. Defaults to the length of the calling string)	boolean
isMatch	Executes a search for a match between a regular expression and the calling string.	pattern: string (regular expression), flags: string (string that contains the flags to add)	boolean
match	Executes a search for a match in a calling string.	pattern: string (regular expression), flags: string (string that contains the flags to add)	string undefined
split	divides the calling string into an ordered list of substrings.	separator: string (pattern describing where each split should occur. The separator can be a simple string or it can be a regular expression)	array<string>
indexOf	Returns the index within the calling string of the first occurrence of the specified value.	searchString: string (The string value to search for) position: number (An integer representing the index at which to start the search)	number (returns -1 if the value is not found)
lastIndexOf	Returns the index within the calling string of the last occurrence of the specified value.	searchString: string (The string value to search for) position: number (The index of the last character in the string to be considered as the beginning of a match)	number (returns -1 if the value is not found)

includes	Performs a case-sensitive search on the calling string to determine whether one string may be found within another string.	value: string (A string to be searched for within the calling string) position: number (The position within the calling string at which to begin searching for the value. (Defaults to 0))	boolean
replace	Replaces all instances of the search value with the replace value.	searchValue: string (A string to be searched for within the calling string) replaceValue: string (A string to replace the search value with)	string
repeat	Constructs and returns a new string which contains the specified number of copies of calling string, concatenated together.	count: number (An integer indicating the number of times to repeat the string)	string
substring	Returns the part of the calling string between the start and end indexes, or to the end of the string.	start: number (The index of the first character to include in the returned substring) end: number (optional) (The index of the last character to exclude from the returned substring)	string
escape	Escapes all markdown special characters in the calling string.	N/A	string
parseDate	Attempts to parse the calling string as a timestamp (returns 0 if unable to) (which can then be formatted by calling formatDate number function – see below).	format: string (the format to try to parse the date using – see Format Specifiers)	number

formatDate	Attempts to parse the calling string as a date (expects ISO 8601 string) (returns the same value if unable to). Returns the formatted date according to the string pattern passed in. To escape characters, wrap them in square brackets (e.g. [MM]).	pattern: string (see Format Specifiers)	string
length	Returns the length of the calling string.	N/A	number

Array Functions

Name	Description	Parameters	Return Value
toTable	Converts the calling array into a table structure.	headers: array<string> (optional) (the column headers to display) templates: array<string> (optional) (expression template strings to use for formatting each cell. Can be used to reference other item properties but not data in the context. Example: 'firstName + ' + lastName toUpperCase()').	string
mapProperty	Converts an array of objects to an array of string values by retrieving a specific property value from each object.	property: string (the property to retrieve from each object in the array)	array<any>
length	Returns the length of the calling array.	N/A	number
includes	Returns true if the value passed is found in the calling array.	value: any (A value to be searched for within the calling array) position: number (The position within the calling string at which to begin searching for the value. (Defaults to 0)	boolean

Object Functions

Name	Description	Parameters	Return Value
length	Returns the length of the keys in the calling object.	N/A	number
toJSON	Converts the calling object to formatted JSON with 2 space indentation.	N/A	string
getPropertyNames	Returns a string array containing all object property names.	N/A	array<string>

Date Functions

Name	Description	Parameters	Return Value
formatDate	Attempts to convert the timestamp as a date (returns the same value if unable to). Returns the formatted date according to the string pattern passed in. To escape characters, wrap them in square brackets (e.g. [MM]).	pattern: string (see Format Specifiers)	string undefined

Global Functions

Name	Description	Parameters	Return Value
formatDate	Returns the current date on the user's machine according to the string pattern passed in. To escape characters, wrap them in square brackets (e.g. [MM]).	pattern: string (see Format Specifiers)	string
now	Returns the current timestamp on the user's machine.	N/A	number
isDefined	Checks to see if a context property is defined.	path: string (the path to check)	boolean

Date Format Specifiers

Format	Output	Description
--------	--------	-------------

YY	18	Two-digit year
YYYY	2018	Four-digit year
M	1-12	The month, beginning at 1
MM	01-12	The month, 2-digits
MMM	Jan-Dec	The abbreviated month name
MMMM	January-December	The full month name
D	1-31	The day of the month
DD	01-31	The day of the month, 2-digits
d	0-6	The day of the week, with Sunday as 0
dd	Su-Sa	The min name of the day of the week
ddd	Sun-Sat	The short name of the day of the week
dddd	Sunday-Saturday	The name of the day of the week
H	0-23	The hour
HH	00-23	The hour, 2-digits
h	1-12	The hour, 12-hour clock
hh	01-12	The hour, 12-hour clock, 2-digits
m	0-59	The minute
mm	00-59	The minute, 2-digits
s	0-59	The second
ss	00-59	The second, 2-digits
SSS	000-999	The millisecond, 3-digits
Z	+05:00	The offset from UTC, ±HH:mm
ZZ	+0500	The offset from UTC, ±HHmm
A	AM PM	

a	am pm	
---	-------	--

Expression Limitations

All expressions (except for expression blocks in Markdown) can be a maximum length of 1000 characters.

Template Limitations

Template names can be no longer than 100 characters.

Template descriptions can be no longer than 200 characters.

The max iteration count for template loops is 1000.

The 'Set Context Property' block property path can be no longer than 200 characters

Expressions can be no longer than 1000 characters.

The total length of Markdown used in the template can be no longer than 100,000 characters.

User Input Limitations

Descriptions can be no longer than 200 characters.

Page titles can be no longer than 200 characters.

Field labels can be no longer than 200 characters.

Field help text can be no longer than 500 characters.

Field context properties can be no longer than 50 characters.

Text field default values can be no longer than 1000 characters.

Text field regular expressions can be no longer than 1000 characters.

Select field default values can be no longer than 1000 characters.

Select field options can be no longer than 2000 characters.

Template Issue Field Administration

Template issue fields define the Jira Issue fields which are available to the expression context (see [Expressions](#)).

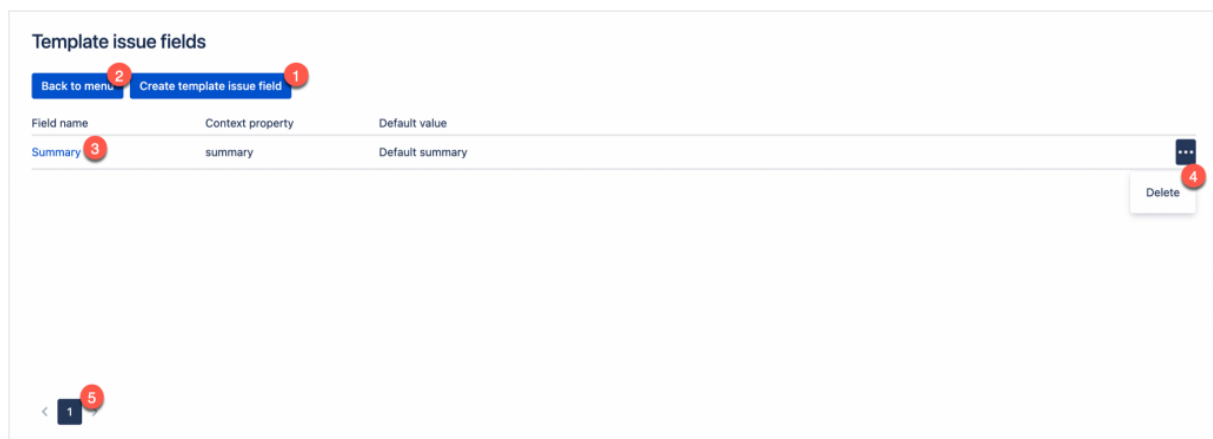
Managing Template Issue Fields

Template issue fields define the Jira Issue fields which are available to the expression context (see [Expressions](#)).

To manage template issue fields, click on the 'Template issue fields' link on the Power Markdown main menu.

Note: Template issue field administration is only available to template administrators.

Note: Template issue field administration is available via the Power Markdown main menu and in the template screen (for quick access to add template issue fields when building templates).



1. Create template issue field – click to open the template issue field modal window enabling creation of a template issue field.
2. Back to menu – click to go back to the main menu.
3. Field name link – click to open the template issue field modal window enabling editing of a template issue field.
4. Delete template issue field – deletes the template issue field after user confirmation.
5. Page template issue fields – click on the page number to navigate to the chosen page.

Creating and Updating Template Issue Fields

The process for creating and updating template issue fields are similar and explained below: Click on the create template issue field button (1) or on the name of the template issue field you wish to edit (2).

Select an issue field from the drop down.

Create template issue field

Issue field *

Select...

Status Category Changed

Fix versions

Last Viewed

Custom URL

Custom UserPicker

Customer

Default Value

A default value is required and is used when the user does not have permission to access the field, where the Jira Project does not display the field, where the value has not been set and in the editor context when designing the template.

The default value input will change depending on the type of the field selected. For example an array type will display a multiline input where each line is an array item.

Most field types support blank string defaults except date, date time and number fields.

Date and date time field types require a date to be entered as a default. This date is converted to a timestamp and formatting functions can be used to convert into date strings within the templates (see [Expressions](#)).

Enter a context property name for the field (the value for this field will be available in the expression context under the issue root property. for example: issue.issueType) The app will auto generate a name based on the field name but this can be changed.

Issue field *

Issue Type

Context property *

issueType

Default value *

Click 'Create' or 'Update'.

Template Issue Field Limitations

Context properties can be no longer than 50 characters.

Default values can be no longer than 1000 characters.

The maximum number of template issue fields is 500.

Variables Administration

Overview

Variables are named and typed pieces of data which are used in templates.

A variable can be created and managed using either the Jira interface or via a secured API endpoint.

Managing Variables

To manage variables, click on the 'Manage Variables' link on the Power Markdown main menu.

The screenshot shows the 'Variables' management interface. At the top, there are two buttons: 'Back to menu' (callout 2) and 'Create variable' (callout 1). Below these are two input fields: 'Name' (callout 3) and 'Filter' (callout 4). A table below lists variables with columns: Name (callout 6), Description, Type, and Created (callout 5). A single row is visible with 'variableName', 'Variable Description', 'Object', and '06/10/2022 19:42:44'. A menu icon (three dots) is next to the row, with 'Copy' (callout 7) and 'Delete' (callout 8) options. At the bottom left, there is a pagination control showing '< 1 >' (callout 9).

1. Create variable – click to navigate to the variable screen to create a variable.
2. Back to menu – click to go back to the main menu.
3. Filter field selection – choose the field you wish to filter on.
4. Filter field value – enter a value to automatically filter the grid based on the selected filter field.
5. Sort grid – click on the name, description, type or created on column headers to sort the rows.
6. Edit variable – click on the variable name to navigate to the variable edit screen.
7. Copy variable – prompts for a variable name and copies the chosen variable. Note: if the variable name is already taken a validation message will be displayed.
8. Delete variable – deletes the variables after user confirmation.
9. Page variables – click on the page number to navigate to the chosen page.

Variable Data Types

A variable's data type determines the values that the variable can contain.

Data types are used to enforce valid data is submitted both in the UI and via the API.

The 'Object' type is flexible and accepts any valid JSON data.
Below is a table listing the available data types and supported values:

Data Type	Description
String	Stores a string value.
Number	Stores any numeric value (6, -7.2, 5, -3.14159).
Boolean	Stores boolean values: true or false.
Date	Stored as timestamps. Accepts dates in string or timestamp format and attempts to convert string dates to timestamps.
Object	Accepts any valid JSON string.

Creating and Editing Variables

Navigate to the variable screen by clicking on the 'Manage Variables' link on the Power Markdown main menu and then the 'Create variable' button above the variables grid (to create) or on an existing variables name (to edit).

To create a variable:

1. Enter a name for the variable (this must not match an existing variable or system context property name and a validation message will be displayed on save if it does).
2. Enter a description (optional)
3. Choose a data type for the variable (see [Variable Data Types](#)).
4. Select or enter a value for the variable.
5. Click 'Save' or 'Save and close'.

The screenshot shows a form for creating or editing a variable. At the top right, it says "Unnamed variable". At the top left, there are three buttons: "Save", "Save and close" (with a red circle containing the number 5), and "Close". The form has four main sections, each with a red circle containing a number: 1. "Name" with a red asterisk and a text input field. 2. "Description" with a text input field. 3. "Type" with a dropdown menu currently showing "String" and a red circle containing the number 3. 4. "Value" with a red asterisk and a large text input field. A red circle containing the number 4 is next to the "Value" label.

Editing a variable is similar to creating a variable however once a variable has been created the variable data type cannot be changed.

To edit a variable:

1. Modify the variables name, description and/or value.
2. Click 'Save' or 'Save and close'.

Managing Variables via the API

The variables API enables programmatic access to create, update, delete and retrieve variables.

Authentication

The variables API uses an API key for authentication.

The API key is available by clicking on the 'Variables API' link in the Power Markdown admin menu (see [API Settings](#)).

The API key should be passed in the 'api-key' header.

Status Codes

The following table lists the possible status codes returned from the API.

Status Code	Description
200 (Ok)	Success response
404 (Not Found)	Error response indicating that the requested resource could not be found.
500 (Internal Server Error)	A server side error occurred.
400 (Bad Request)	The request is incorrect or corrupt.
401 (Unauthorised)	The API key provided is invalid.

Error Responses

Where the status code does not indicate success an error response body is returned containing an error code and an optional message.

```
{  
  errorCode: 100,  
  message: 'Error message'  
}
```

Error Code	Description
100 (Invalid HTTP Method)	The HTTP method is not supported.

101 (API Key Not Found)	The API key has not been provided in the headers.
102 (Invalid API Key)	The provided API key does not match the active API key.
103 (Name Not Found)	The name query string parameter has not been provided.
104 (Variable Not Found)	The variable requested cannot be found.
105 (Max Variable Count Reached)	The maximum count of variables has been reached.
106 (Invalid Body)	The body passed in the request is not in the expected format.
107 (Variable Value Exceeds Max Size)	The variable value provided is larger than the maximum allowed.
108 (Variable Already Exists)	The variable already exists and cannot be created.

Create Variable

Method	POST
Body	<pre>{ "name": string, "description": string, "type": "string" "number" "date" "boolean" "object", "value": any }</pre>
Example body	<pre>{ "name": "variable1", "description": "Variable 1 description", "type": "string", "value": "value" }</pre>
Success status code	OK (200)

Possible error responses	Bad request (400) – Invalid body (106) Bad request (400) – Variable value too large (107) Bad request (400) – Max variable count reached (105) Bad request (400) – Variable already exists (108)
--------------------------	---

Update Variable

Method	PATCH
Notes	The body must include the name and then can include either the description, value or both.
Body	<pre>{ "name": string (required), "description": string (optional), "value": any (optional) }</pre>
Example body	<pre>{ "name": "variable1", "value": "value" }</pre>
Success status code	OK (200)
Possible error responses	Bad request (400) – Invalid body (106) Bad request (400) – Variable value too large (107) Not found (404) – Variable not found (104)

Delete Variable

Method	DELETE
Query string	?name=VARIABLE_NAME
Success status code	OK (200)

Possible error responses	Bad request (400) – Name not found (103) Not found (404) – Variable not found (104)
--------------------------	--

Get Variable

Method	GET
Query string	?name=VARIABLE_NAME
Success status code	OK (200)
Possible error responses	Bad request (400) – Name not found (103) Not found (404) – Variable not found (104)
Response body	<pre>{ "name": "variableName", "description": "Variable description", "type": "string", "value": "..." }</pre>

Get All Variables

Method	GET
Response body	<pre>[{ "name": "variableName", "description": "Variable description", "type": "string" }]</pre>
Success status code	OK (200)

Variable Limitations

Variable names can be a maximum length of 50 characters.

Variable descriptions can be a maximum length of 100 characters.

Variables can be a maximum size of 500kb.

No more than 200 variables can be created.

Using Markdown Templates

There are two ways to use a markdown template:

1. Apply a template to an issue. This enables the end user to choose a template and an issue and either update an existing field (which supports wiki markup) or create an issue comment. The field is then updated or the comment is created and does not change over time (the generated content is static).
2. Dynamic markdown. Display content generated in real time in the issue (either in a panel in the main issue screen (below the description) or in an activity tab (along with comments, history etc.)).

The following sections describe using these methods in more detail.

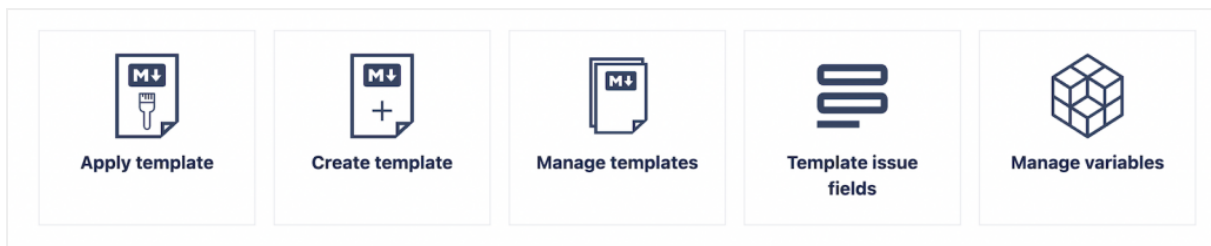
Applying to an Issue

To apply a Markdown template to an issue in order to create a comment or update the description (or other field which supports wiki markup) follow the steps below.

Note: It is possible to apply a template from the Power Markdown main menu and also from the issue screen via the issue actions menu ('...' menu in the top right). Both options use the same process with the omission of the issue selection when applying from the issue screen. Both approaches are described below.

A: When applying a Markdown template from the Power Markdown main menu:

Navigate to the Power Markdown main menu and click 'Apply Template'.



Select or search for an issue to apply the template to and click next.

Apply Template

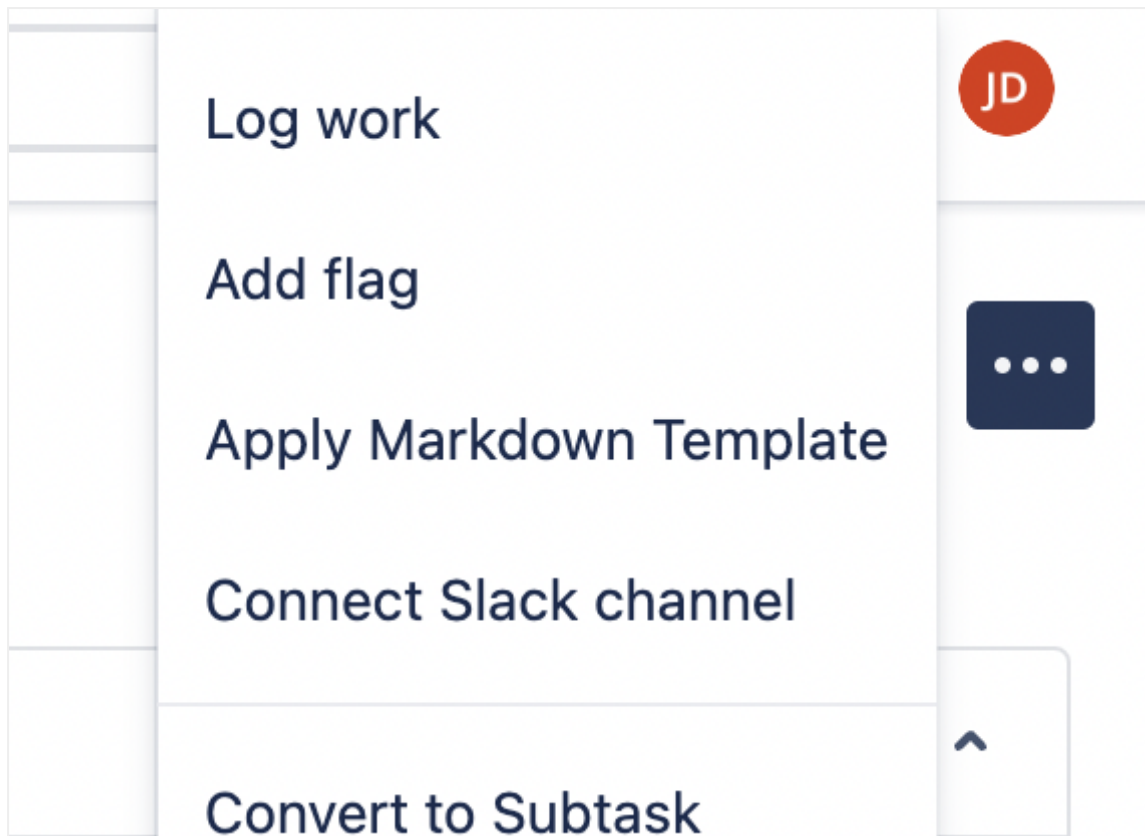
Please select or search for the issue you wish to apply the template to.

Issue

- FH-2 Hotel Reservation
- FH-1 Hotel search

Cancel Next

B: When applying a Markdown template from the issue screen:
Navigate to the issue you wish to apply a template to and from the issue actions menu in the top right corner '...' select 'Apply Markdown Template'.



From this point forward all steps apply to both A and B above.

Select a template to use and click next. The templates visible will depend on:

1. The users roles and the roles associated with templates.
2. The template display condition (if set).

Click next.

Apply Template

Please select the template you wish to use.

Template

Feature Test Results

- Feature Test Results
- Client Details

Cancel Back Next

Select the template target:

1. If the template supports field selection only then you will be prompted to select a field.
2. If the template supports field selection and the creation of comments, then you will be prompted to select a field or whether to create a new comment.
3. If the template only supports comment creation then selection will be skipped and you will be directed to either the preview or user input (if required).

Click next.

Apply Template

Please select a target for the selected template.

Target

Description

Description

New comment

Cancel

Back

Next

If the template has any user input defined, the user input pages will be displayed. Navigate through the user input pages until the end and click next.

Apply Template

General

Tester name *

Ethen Rojas

Passed? *



Test Script *

https://

Cancel

Back

Next

Depending on the template preview settings you will then either be displayed with a read-only preview or will be able to edit the generated content.
Click apply.

Apply Template

Review and make any changes to the generated content below. When you are ready you can apply to the issue by clicking 'Apply'.

Normal text ▾ | **B** *I* ... | A ▾ | ☰ ☷ | 🔗 @ 🗃 <> ⓘ ” — + ▾

Feature Test Results

Test Script

✔ **Status: Passed**

Notes:

Great work!

It would be good to display directions to the hotel, I'll open another ticket.

Cancel

Back

Apply

Preview with edit

Apply Template

Review the generated content below. When you are ready you can apply to the issue by clicking 'Apply'.

Feature Test Results

Test Script

✓ Status: Passed

Notes:

Great work!

It would be good to display directions to the hotel, I'll open another ticket.

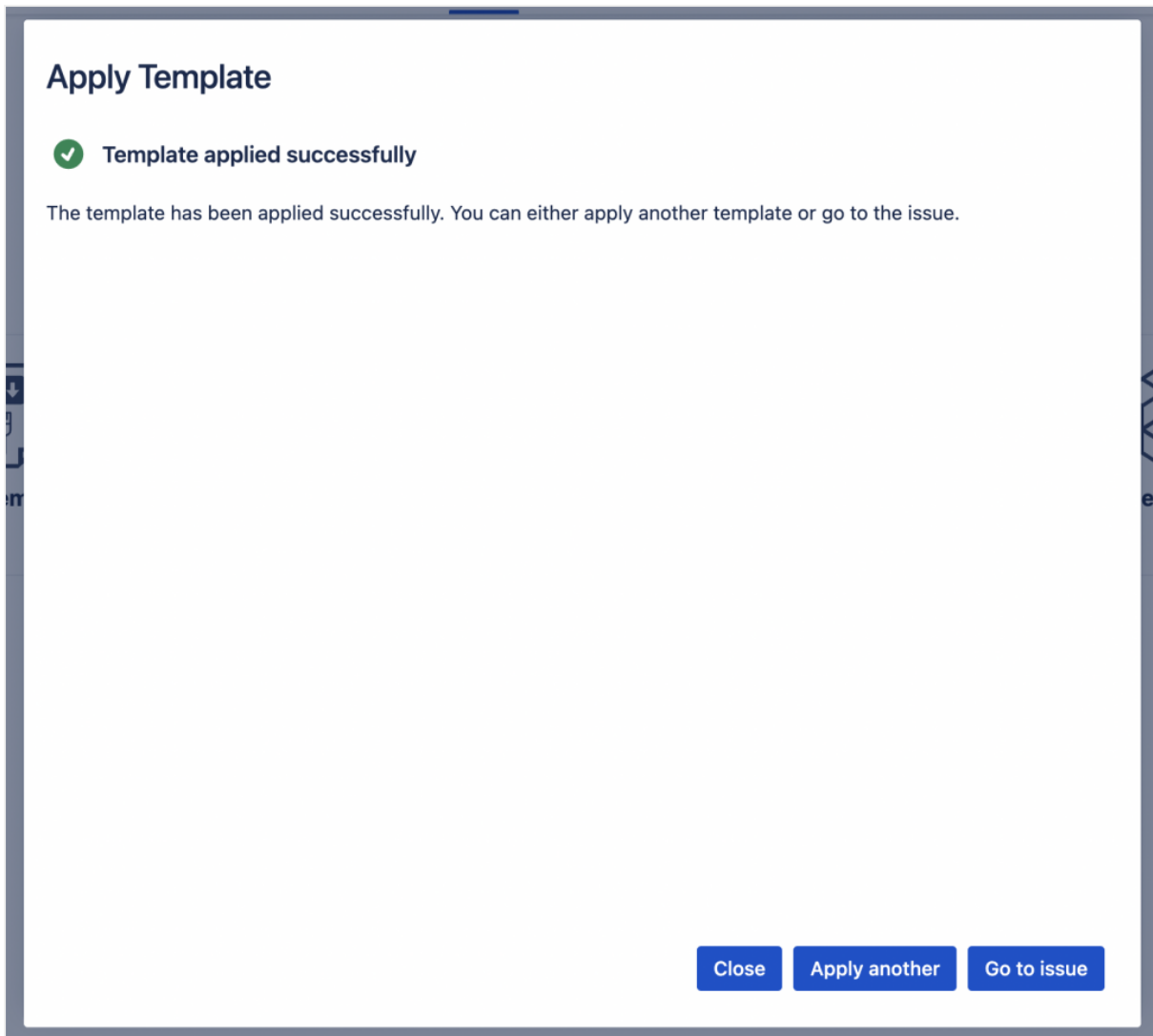
Cancel

Back

Apply

Preview – Read only

Then either click the 'Go to issue' or 'Refresh issue' button (depending on whether the Markdown template is being applied from the main menu or within an issue) to navigate to the chosen issue or the 'Apply another' button, to apply another template.



Dynamic Issue Markdown

Dynamic Markdown enables content to be generated in real time in the issue screen (either in a panel in the main screen (below the description) or in an activity tab (alongside comments and history)).

Both sections are identical in function and allow viewing the same content but each can have its own selection. The panel in the main screen can provide greater visibility whereas the activity tab has more space.

Creating and updating Dynamic Markdown follows a similar process. The only difference is that once a visibility is selected it cannot be modified.

Choose whether to display the Dynamic Markdown in the issue panel or activity tab.

1. If the issue panel is not already open, click on the Power Markdown icon in the issue menu.
2. For the activity tab, click on the Power Markdown tab in the activity tabs.

Projects / Fly Hotels / FH-1

Hotel search

Attach Create subtask Link issue Power Markdown Add Checklist

Description
Add a description...

Power Markdown
Client Details Display all

Brown-Veum

IN CONTRACT

License Level
PLATINUM

Address
9 Market Place, Thetford, Norfolk County,
IP24 2AL

Telephone

JD Add a comment...
Pro tip: press M to comment

To Do

Details

Assignee Unassigned
Assign to me

Reporter JD John Dodson

Labels None

Client Brown-Veum

Priority Medium

More fields Story Points, Original estimate, Time ...

Created 5 November 2022 at 09:05
Updated 25 seconds ago

Dynamic Markdown in issue panel

Projects / Fly Hotels / FH-1

Hotel search

Attach Create subtask Link issue Power Markdown Add Checklist

Description
Add a description...

Activity
Show: All Comments History Work log Power Markdown Checklist history Newest first

Client Details Display all

Brown-Veum

IN CONTRACT

License Level
PLATINUM

Address
9 Market Place, Thetford, Norfolk County,
IP24 2AL

Telephone
01842 753018

To Do

Details

Assignee Unassigned
Assign to me

Reporter JD John Dodson

Labels None

Client Brown-Veum

Priority Medium

More fields Story Points, Original estimate, Time ...

Created 5 November 2022 at 09:05
Updated 25 seconds ago

Dynamic Markdown in activity panel

If no Dynamic Markdown is available then a message informs you of this.

Select... Display all

No dynamic markdown exists currently. To create, click on the plus icon above.

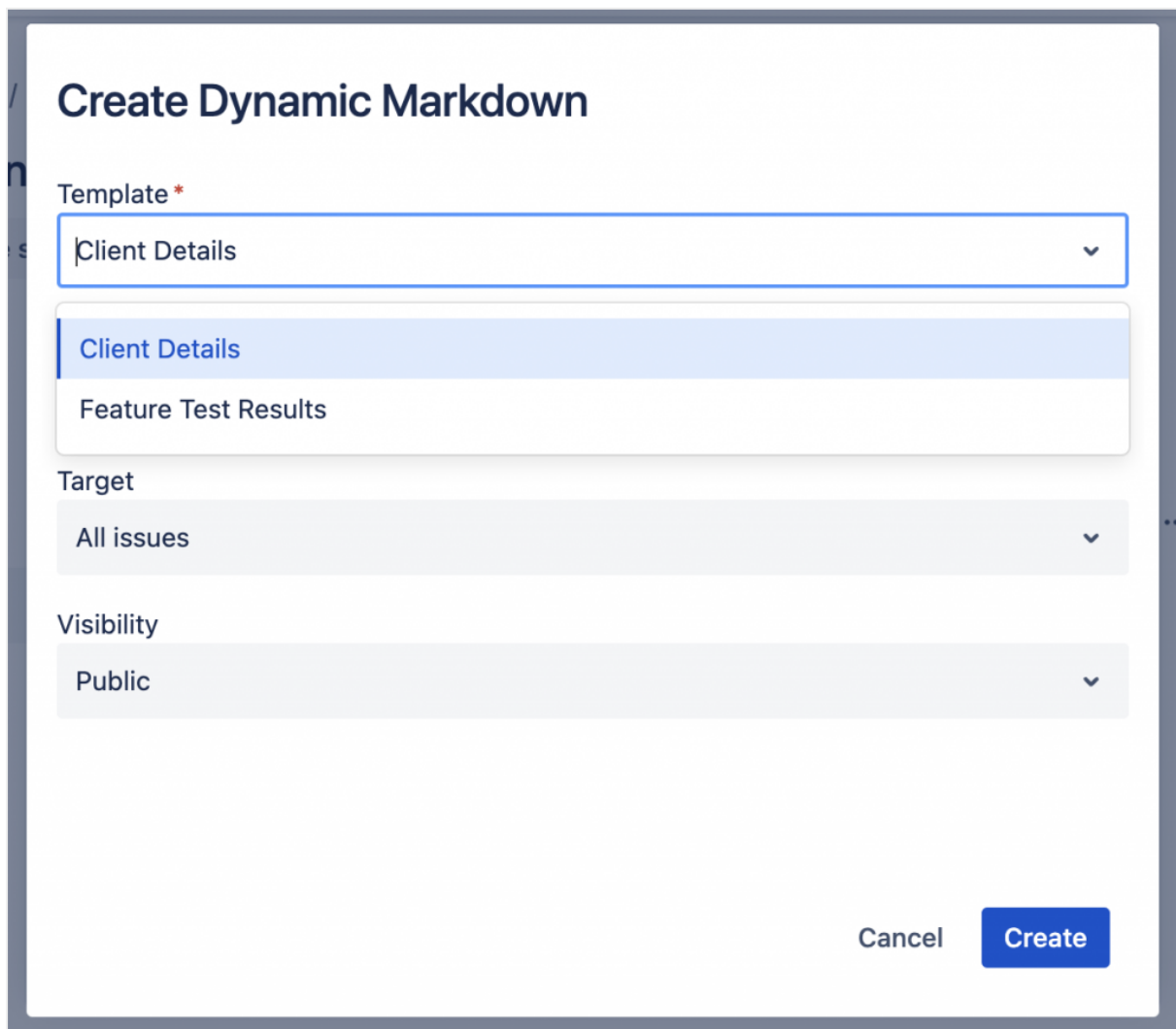
Click on the create (plus icon) (1) or edit (pencil icon) (2) (when Dynamic Markdown has been selected) which will open the Dynamic Markdown modal window.



Select a template. The templates visible will depend on:

1. The users roles and the roles associated with templates.
2. The display condition (if set).

Enter a title (this is displayed above the generated content).



Select a target:

1. All issues – this will display the Dynamic Markdown for all issues in Jira.
2. Current issue – this will display the Dynamic Markdown only for the current issue.

3. Conditional – this will display the Dynamic Markdown based on the result of the condition expression. The condition expression has access to the template issue fields in the expression context and displays the Dynamic Markdown if the condition evaluates to a truthy value (false, undefined or null equate to false all other values equate to true).

Create Dynamic Markdown

Template *
Client Details

Title *
Client Details

Target
All issues

- Current issue
- All issues
- Conditional

Cancel Create

Select the visibility (only available on creation of the Dynamic Markdown and only for template administrators. For other users only private is permitted and the option is not displayed):

1. Public will display the Dynamic Markdown for all users.
2. Private will only display the Dynamic Markdown for just the user creating it.

Create Dynamic Markdown

Template *
Client Details

Title *
Client Details

Target
All issues

Visibility
Public

Public
Private

Cancel Create

If the selected template requires input data then a 'Next' button will appear instead of 'Create' and you are prompted to enter input data until a complete page is displayed at which point the 'Create' button is displayed.

Click 'Create' or 'Update'.

Note: The Dynamic Markdown is displayed in the select list in alphabetical order.

Deleting Dynamic Markdown

1. Select the Dynamic Markdown you wish to delete.
2. Click the trash icon.
3. Confirm the request to delete.
4. The markdown will now be deleted.

Setting Default Dynamic Markdown

You can choose Dynamic Markdown to set as default for all issues which meet the target criteria.

To do this click the 'Set as default' button (which will display a green tick when the default is set).

To remove the default click the button again and the green tick will be removed.

Client Details ▼ + ✎ 🗑️ ○ ⊗ Display all

Brown-Veum

Set as default

Default not selected

Client Details ▼ + ✎ 🗑️ ✓ ⊗ Display all

Brown-Veum

Default selected

Overriding Dynamic Markdown Target Settings

It is possible to view all Dynamic Markdown regardless of target settings.

To do so click on the 'Display all' toggle button.

When the 'Display all' is toggled on, all Dynamic Markdown will be displayed in the select list regardless of target settings. This enables modification of the Dynamic Markdown.

Client Details ▼ + ✎ 🗑️ ✓ ⊗ Display all

Brown-Veum

Dynamic Markdown Limitations

The dynamic Markdown title can be a maximum length of 200 characters.

For templates which have user input, the maximum length of a text field is 5000 characters.

Administration

User Roles

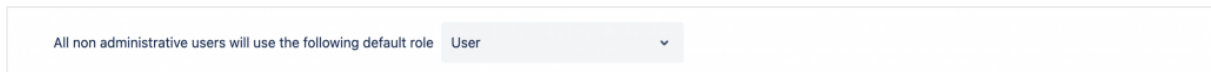
The user roles screen enables the creation of custom roles and provides the ability to assign roles to Jira users.

Roles have two purposes:

1. They control which users can administer markdown templates and variables.
2. They can control which markdown templates users can see and use (when creating templates, the template administrator can choose which roles to make that template visible to, thereby restricting access to templates based on roles).

Assigning a Default Role

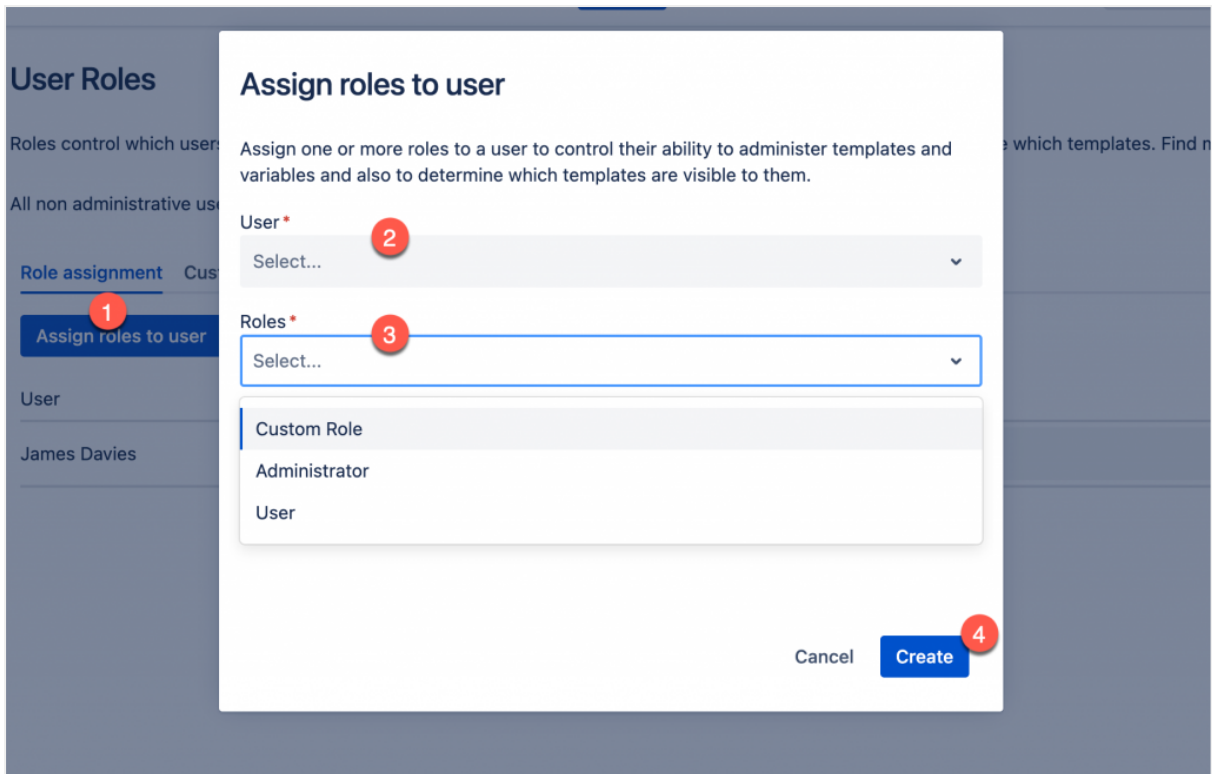
It is possible to assign a default role for all non administrative users. By default that role is the 'User' role (which does not have rights to manage templates or variables).

A screenshot of a user interface element showing a default role selection. It consists of a light gray rectangular box. On the left side of the box, the text "All non administrative users will use the following default role" is displayed in a small, dark font. To the right of this text is a dropdown menu. The dropdown menu is currently open, showing the word "User" in a dark font. To the right of the word "User" is a small, dark downward-pointing arrow icon.

Assigning Roles to a User

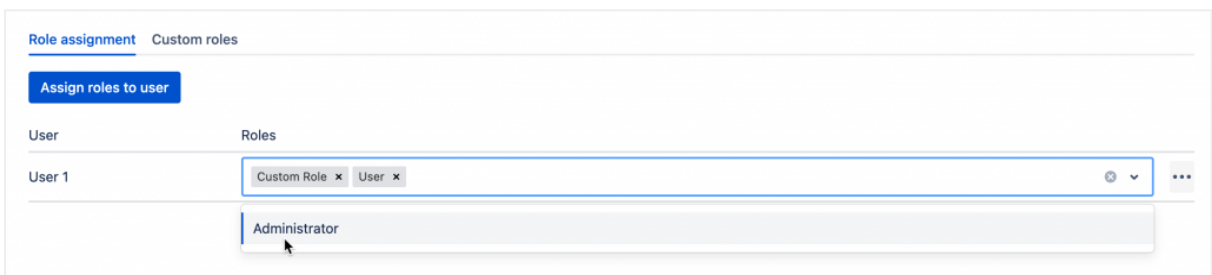
If it is preferred to assign roles to specific users you can do so in the role assignment tab.

1. Click 'Assign roles to user'.
2. Choose the user you wish to assign roles to.
3. Choose one or more roles you wish to assign to the user.
4. Click 'Create'.



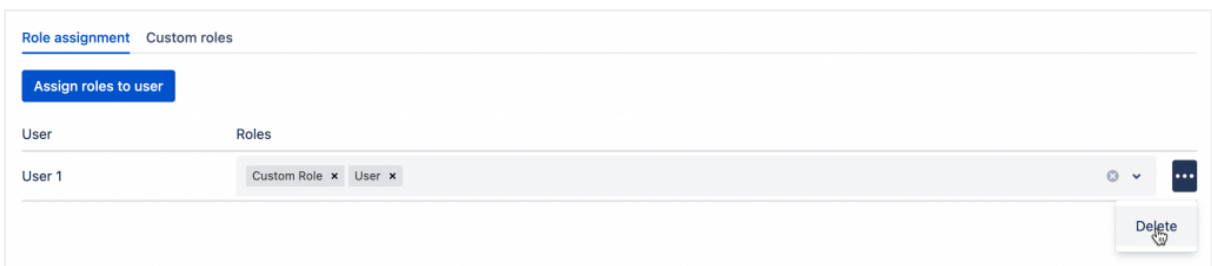
Changing a Users Roles

In the role assignment tab, it is possible to update a user's roles by adding or removing roles using the roles drop down.



Deleting User Roles

In the role assignment tab, it is possible to remove assigned roles for a user by clicking on the '...' button in the grid and selecting 'Delete'. After confirming you wish to delete, the assigned roles will be removed. This user will then fallback to using the default role.



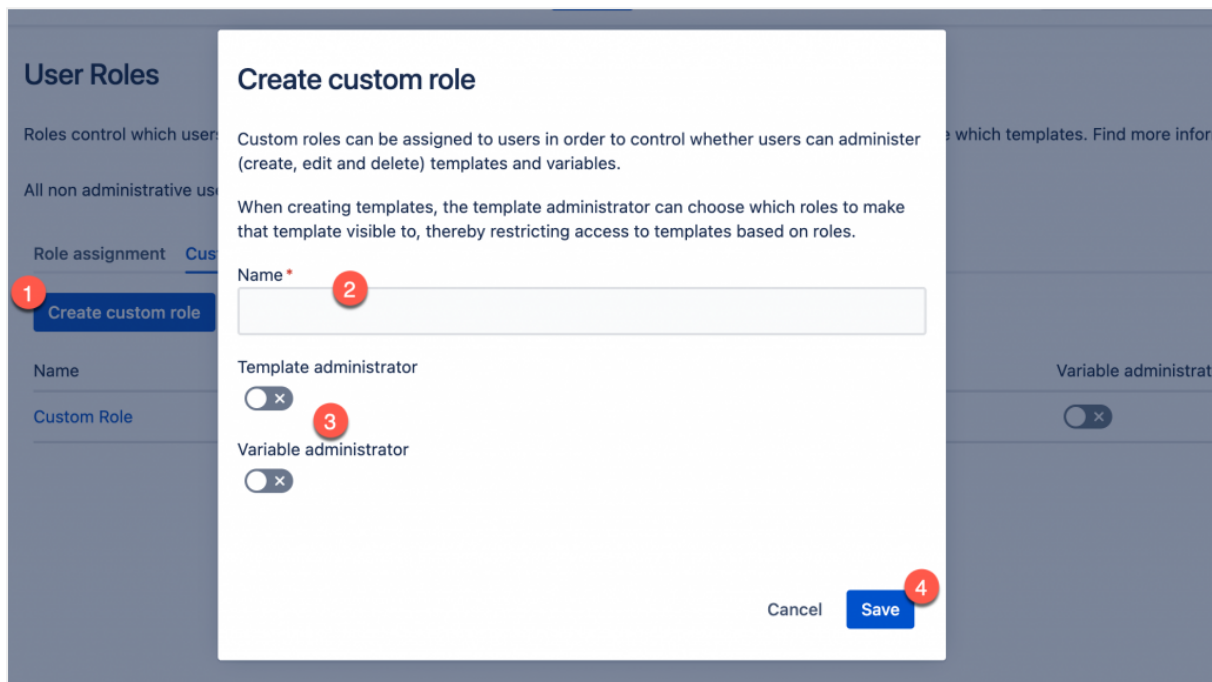
Creating a Custom Role

In the custom roles tab it is possible to create custom roles to be used to:

1. Control which users can administer templates or variables.
2. Assign to users to create groups of users which can use specific templates.

To create a custom role:

1. Click 'Create custom role'.
2. Enter a name for the role.
3. Choose whether the role supports template or variable administration.
4. Click 'Save'.



The screenshot shows a 'Create custom role' dialog box overlaid on a 'User Roles' page. The dialog box contains the following elements:

- Title:** 'Create custom role'
- Text:** 'Custom roles can be assigned to users in order to control whether users can administer (create, edit and delete) templates and variables.' and 'When creating templates, the template administrator can choose which roles to make that template visible to, thereby restricting access to templates based on roles.'
- Name field:** A text input field with a red '2' indicating where to enter the role name.
- Template administrator:** A toggle switch with a red '3' indicating where to click to enable this role.
- Variable administrator:** A toggle switch.
- Buttons:** 'Cancel' and 'Save' (with a red '4' indicating where to click).

Updating a Custom Role

A custom role can be updated by following the steps below:

1. Click on the name of the role.
2. Make updates to the roles name and/or template/variable administration status.
3. Click 'Update'.

Alternatively you can change the custom roles variable and template administration status by toggling the switches in the grid.

Custom Role Limitations

Custom role names can be a maximum length of 100 characters.

Variables API

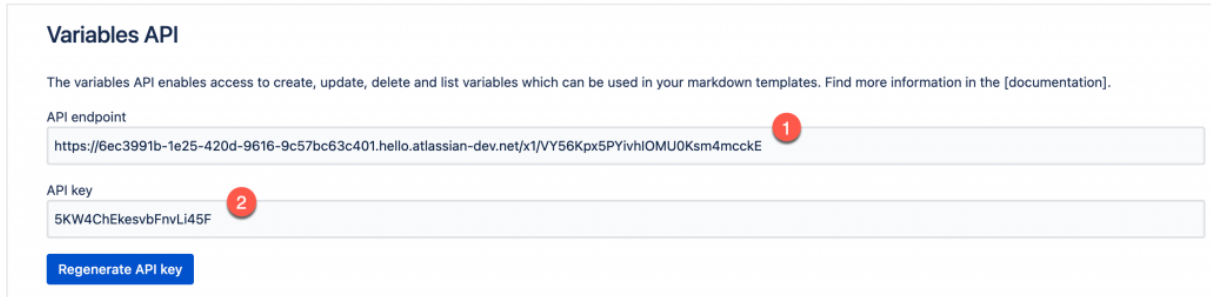
The variables API enables programmatic access to create, update, delete and list variables. The admin section provides the URL endpoint required for access and control over the API key used to secure the endpoint.

API Settings

To use the variables API it is necessary to use the API endpoint and the API key.

For more information about using the variables API see [Managing Variables via the API](#)

The API endpoint (1) and key (2) are made available via the 'Variables API' link in the admin menu (see [Admin Menu](#)).



Variables API

The variables API enables access to create, update, delete and list variables which can be used in your markdown templates. Find more information in the [documentation].

API endpoint 1

https://6ec3991b-1e25-420d-9616-9c57bc63c401.hello.atlassian-dev.net/x1/VY56Kpx5PYivhiOMU0Ksm4mckE

API key 2

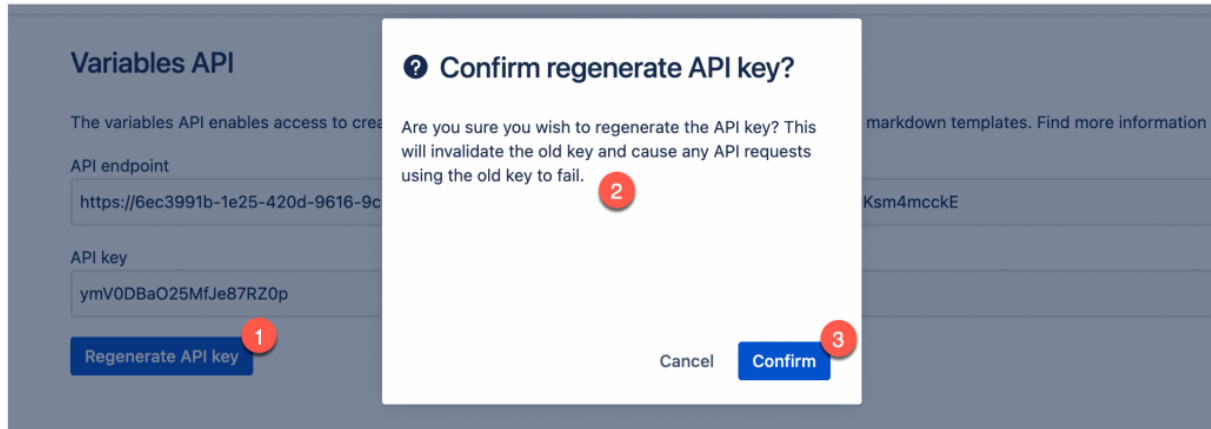
5KW4ChEkesvbFvnLi45F

[Regenerate API key](#)

Regenerating the API Key

If it is necessary to change your API key it is possible to regenerate it by following the steps below:

1. Click on the 'Regenerate API key' button.
2. Review the warning message (which is letting you know that any requests to the endpoint using the old API key will fail).
3. Confirm the warning.



Variables API

The variables API enables access to create, update, delete and list variables which can be used in your markdown templates. Find more information in the [documentation].

API endpoint

https://6ec3991b-1e25-420d-9616-9c57bc63c401.hello.atlassian-dev.net/x1/VY56Kpx5PYivhiOMU0Ksm4mckE

API key

ymV0DBaO25MfJe87RZ0p

[Regenerate API key](#) 1

Confirm regenerate API key?

Are you sure you wish to regenerate the API key? This will invalidate the old key and cause any API requests using the old key to fail. 2

[Cancel](#) [Confirm](#) 3