Versions overview

Last Modified on 04/18/2024 11:42 am EDT

Your knowledge base content is subject to change. Perhaps your internal policies have been updated or the software you are documenting has undergone a major change to its interface.

You will need to update your knowledge base content and it can be really daunting to do those edits in the already published article.

You can create specific, distinct versions of an article or custom content category over time, to reflect changes to the content. Versions are basically a permanent snapshot you create of the article at a given time. Only one version of an article is published in your knowledge base at a time.

With versions, you can:

- Create different **Versions** of your articles (minor, major, or custom numbering).
- Make updates/edits in the new version rather than the currently published version.
- Activate the new version in place of your existing article once the version is ready to go.
- Add editorial notes for each version to summarize what has changed or provide editorial feedback.

Benefits of using versions

Some of the benefits of using versions include:

- New versions begin in an inactive status, so you can make as many edits as you want to that version instead of the currently published article. This means your readers never stumble across an in-progress edit.
- Versions allow you to create an updated article ahead of time and then publish the new version to coincide with a software release or policy update.
- Older versions can serve as back-ups and will allow you to easily reference historical content--particularly helpful if you accidentally removed a section you still need or need to provide an audit history of previous documentation versions.
- Unlike revisions, versions are stored for as long as you'd like them to be.
- You can add version notes to summarize changes that were made or track content reviews.
- You can create versions for both articles and custom content categories.

Use cases for versions

Generally, people use versions to:

- Keep an audit history of different versions of content used over time
- Provide version control for their documentation
- Match documentation updates to real-world updates (such as releases, etc.)

- Prepare updates for published content without having to instantly publish them

Do not use versions for:

- Tracking every tiny change to an article
- Tracking every person who's edited an article over time

For Version Control: When you create a new version, it takes a copy of the current active version to use as a starting point. Any edits you make to one version don't impact another.

For releasing updates to an already-published article when you choose: You can also wait to publish a new version until all editing and review is done--this can be a great way to prep documentation updates ahead of major company announcements, rebranding, product releases, a new school year, etc. without impacting the current published article.

For allowing a review of updates to a published article before they are published: Since you can also control which authors can create versions and who can publish them, versions can be a great way to allow folks to edit and update articles without giving them access to edit a currently-published article.

At this time, we do not have an automatic version comparison that will highlight the differences between versions, like we do for revisions. For now, you'd need to manually compare versions or have detailed Version Notes highlighting the differences.

Are revisions stored for versions?

We were hoping you wouldn't ask this, as it's not an easy question to answer. The short answer is: yes, revisions are automatically stored for *some* versions. But the long answer is: it's complicated.

We track the ten most recent revisions to an article automatically. When you create an article, that is automatically created as Version 1.00. As long as you continue with Version 1.00, revisions will be tracked on that version.

Once you create more than one version, we track revisions for **the currently active version only.**

After a version is deactivated, it continues to hold the revisions it had from when it was activated (max of 10).

Let's look at an example to see how this plays out:

Linus has an article called "Learning to Fly" which he has published. By default, this has only one version: **Version 1.0.**

- Revisions for Version 1.0 are tracked, up to a maximum of 10.

Linus needs to make some updates, so he creates a new minor version, **Version 1.1.**

- Revisions for Version 1.0 will still be tracked, since it is still the current active version.
- Revisions for Version 1.1 will not be tracked because it isn't activated yet.

Linus continues to save changes to **Version 1.1** but has not yet activated it.

- Those changes will **not** be tracked as revisions, because Version 1.1 has not yet been activated.

- Any revisions to **Version 1.0** would continue to be tracked.

Linus activates **Version 1.1**.

- From activation onward, revisions for Version 1.1 will be tracked.
- No further revisions for Version 1.0 will be tracked.
- The historical record of the last 10 revisions for Version 1.0 will still appear if we view Version 1.0 in the editor; they will not appear when we view Version 1.1 in the editor.