ATLORIA

Preparing a Version for Final Release Review

August 25, 2026

Confirming the version is ready for release review

In Atloria, start from your project workspace and open the **Versions** area, then select the version you plan to send for final release review. Before you do anything else, make sure you are looking at the correct release candidate. Check the version name, any visible status label, and the surrounding version details so you do not prepare the wrong version by mistake. If your team works with several active versions at once, this quick check prevents comments and approvals from being attached to the wrong release.

Once you are on the version record, review the overall readiness details shown on that screen. Look for checklist-style fields or status indicators tied to **content completeness**, **screenshot coverage**, **comparison status**, and **visibility**. These are the main signals reviewers will rely on when deciding whether the version is ready for signoff. If one of these areas still shows as incomplete, open that section from the version and finish it before requesting approval.

You should also confirm that the content included in this version is no longer sitting in an unfinished state. A version that still contains pages marked as draft, in progress, blocked, or waiting for review is usually not ready for final release review. If you recently worked through comments and status updates, use Understanding Review Statuses Comments and Next Steps as your reference instead of repeating that review here.

Finally, make sure the actions you need are available on the version screen. If you do not see the button or menu option used to move the version forward, you may not have permission to submit it for final signoff. In that case, stop before making other changes and confirm who on your team is responsible for the final submission.

[SCREENSHOT: Version details screen showing version name, current status, readiness fields, and final review action]

Checking that all required content is complete

From the version record, open the content list linked to that version and scan every page included in the release. This is where you confirm that the version contains the full set of documentation promised for the release, not just the pages that were recently edited. Pay close attention to status labels next to each item. If any page still appears as **Draft**, **In Progress**, **Blocked**, **Rejected**, or otherwise unfinished, it needs attention before the version can move into final review.

Use the list to spot gaps that are easy to miss during day-to-day editing. Missing pages often show up as expected topics that are not present at all, while incomplete work usually appears as pages without a completed review state or without a clear owner. If Atloria shows ownership or assignment details in the version content list, verify that every required page has someone responsible for it. Unowned pages are a common reason release review stalls.

As you review the included pages, confirm that the version contains the document types your team expects for that release. Depending on your project, that may include updated feature pages, setup instructions, release notes, or supporting guidance for changed workflows. The key point is consistency: if the release introduces something new or changes an existing workflow, the matching documentation should already be present in the version.

When you find a gap, fix it from the version view instead of pushing the version ahead and hoping reviewers will catch it later. Open the affected page, complete the missing work, and return to the version list to verify the status has updated. If you need a broader refresher on how versions are organized across the release cycle, see Managing Documentation Versions Across the Release Cycle.

[SCREENSHOT: Version content list with page titles, status labels, and ownership details]

Reviewing screenshots and visual updates

If your version includes pages that rely on interface images, open the screenshot review area connected to the version and look for any pages marked as needing new or refreshed visuals. This is especially important when the release changes navigation, button labels, page layouts, or visible controls. A page can be textually correct and still be misleading if the screenshot shows an older screen.

Review each flagged page with the current Atloria content side by side. Check whether the screenshot still matches what readers will actually see. Focus on practical details such as updated menu names, changed section titles, renamed buttons, and

any visible panels or tabs that have moved. If a guide tells readers to click a specific control, the image should show that same control clearly. When the wording or layout no longer matches, replace the image before marking the page as ready.

Also confirm that every page expected to include a screenshot actually has one attached or linked in the version. Missing visuals are easy to overlook when the text is complete, so use the screenshot-required markers in the version review area as your source of truth. If a page is flagged for a screenshot and no image is present, treat that as an incomplete item.

Only mark screenshot review as complete when outdated images have been replaced and all required visuals are present. Do not use the completion marker as a reminder to come back later; approvers will assume that a completed screenshot check means the visual review is finished. If your team manages screenshots across multiple releases, Checking Screenshot Readiness Before Version Release can help you verify that nothing is missing.

[SCREENSHOT: Screenshot review area showing pages that need updated visuals and completion status]

Validating comparisons and change tracking

Before sending a version for final release review, open the comparison view for the current version and compare it with the previous released version. This step helps you confirm that Atloria is showing the release changes clearly and that the documentation matches what actually changed. Reviewers often rely on this view to understand the scope of the release without opening every page one by one.

As you move through the comparison, check how each topic is labeled. Changed pages should be easy to recognize as **new**, **updated**, **unchanged**, or **removed**. Make sure those labels match your expectations. For example, a newly added feature page should not appear unchanged, and a page that was intentionally retired should not still look active in the release package. If the comparison view highlights something unexpected, open the related page and confirm whether the content or version assignment needs correction.

This is also the point where you verify that major product changes have matching documentation updates. If the release includes a visible workflow change, new setup option, or updated navigation path, the comparison should show corresponding page updates. When an expected change is missing, it usually means one of two things:

the documentation was not updated, or the page was updated outside the target version and is not included correctly.

After you finish the review, update the comparison-related fields or completion markers on the version record so approvers can see that the release delta has already been checked. That visible confirmation saves time during signoff and reduces back-and-forth questions about what changed. For a deeper walkthrough of comparison work, refer to Comparing Documentation Versions for Release Decisions.

[SCREENSHOT: Version comparison view showing new, updated, unchanged, and removed topics]

Deciding what will be visible at release

Final release review is not only about what is written; it is also about what readers will actually be able to see when the version goes live. On the version record, review the visibility settings for each included page and confirm whether it should be **published**, **hidden**, or **deferred** from the release. This is where you make sure the release scope matches the real product scope.

Start by checking for pages that should not be part of the public or reader-facing release yet. Draft-only material, internal notes, and content tied to unfinished features should be excluded before final signoff. If a page describes a feature that is still delayed, keep that page hidden rather than letting it slip into the release package by accident. The same applies to content that is still under internal review even if the rest of the version is ready.

Then look at the version as a whole. The version-level visibility decision should match the intended audience and release plan. If the release is meant for a specific audience, confirm that the included pages and their visibility choices support that scope. A mismatch between page-level settings and version-level release intent can create confusion during publishing and review.

This is also a good moment to double-check that no internal-only content remains visible simply because it was copied forward from an earlier version. Reviewers expect the final release candidate to reflect deliberate choices, not leftovers from previous work. If you need more detail on access and visibility controls, see Managing Version Visibility and Reader Access and Validating Version Access Before Sharing or Export.

[SCREENSHOT: Version visibility settings showing page-level publish, hide, and defer options]

Submitting the version for final signoff and fixing common blockers

When all readiness areas are complete, return to the version record and use the available version action to request **final review** or **signoff**. Before clicking it, do one last pass over the visible checklist items on the page. If **content completeness**, **screenshot review**, **comparison review**, or **visibility** still shows as incomplete, Atloria may block submission or send the version forward with obvious gaps that approvers will reject.

If the submission action is unavailable or fails, check the version record for the most common blockers:

- **Incomplete content** in the version content list
- **Missing or outdated screenshots** on pages marked as requiring visuals
- **Unfinished comparison review** between the current and previous release
- **Unset visibility decisions** for pages or the version itself

Fix these issues directly from the linked areas on the version screen, then return and try the submission again. Keeping all corrections tied to the version record makes it easier for the team to follow what changed before resubmission.

If approvers send the version back, open the version comments and review notes attached to the record. Focus on the flagged pages or unresolved release concerns, make the requested updates, and then resubmit the same version once the issues are cleared. You do not need to restart the entire review process if the feedback is limited to a few pages, but you should make sure the status and checklist fields reflect the updated state before sending it forward again.

After submission, confirm that the version status changes to the expected **final review** or **approval** state. That status change is the clearest sign that the version is now in the signoff queue and visible to the rest of the team. The next step is Managing Version Review Requests and Decisions.

[SCREENSHOT: Version record showing final review action, checklist completion, comments, and updated review status]

Overview

Preparing a version for final release review in Atloria means checking the version record from a release perspective rather than an editing perspective. At this stage, you are no longer asking whether a single page looks good on its own. You are confirming

that the full version is complete, visually current, properly compared against the last release, and set up with the right visibility before approvers are asked to sign off.

The core areas to review are:

- **Version identity** — confirm you opened the correct release candidate
- **Content completeness** — make sure all required pages are present and finished
- **Screenshot coverage** — replace outdated visuals and confirm required images exist
- **Comparison status** — verify what changed since the previous release
- **Visibility decisions** — ensure only the right pages will be visible at release
- **Submission readiness** — confirm the version can move into final review without blockers

This preparation work is important because approvers usually expect a version in final review to be nearly release-ready. If obvious issues remain, the review cycle slows down and comments become harder to track. A clean version record with completed review fields gives everyone the same picture of release readiness.

You do not need to repeat the earlier review-status work covered in Understanding Review Statuses Comments and Next Steps. Instead, use this stage to pull those earlier checks together into one final release candidate review. Think of the version record as your control center: if the content list, screenshot review, comparison view, visibility settings, and status actions all line up, the version is ready to move forward.

Prerequisites

Before you prepare a version for final release review in Atloria, make sure the basic release work is already in place. This task assumes you are working with an existing version inside a project workspace and that the version already contains the pages intended for the release. If the version is still being assembled or generated, finish that work first.

You should already have:

- Access to the correct **project workspace**
- A target entry in the **Versions** area
- A version that is far enough along to be treated as a release candidate
- Content pages added to the version and reviewed by the people responsible for them

- Updated screenshots available for any pages that require visuals
- A previous released version available for comparison, if your team uses release-to-release comparisons
- Permission to change the version status or submit it for review

It also helps if you are already comfortable moving between the version record and related areas such as page lists, screenshot review, and comparison screens. If you need help finding your way around project and version navigation, use Managing Project Version Workspaces and Managing Version Lists Statuses and Comparisons before continuing.

If your team uses formal approvals, make sure you know who the final approvers are and what level of completion they expect before a version enters signoff. Atloria can show the version status and review actions clearly, but it is still your job to ensure the release candidate is complete enough that those actions make sense. A version that still has open gaps, hidden uncertainty, or missing release decisions is not ready for final review.