ATLORIA

Using Code Parsing Results To Improve Technical Documentatio

August 25, 2026

Reviewing the parsing outputs that feed documentation

In Atloria, the parsing workspace gives documentation teams a source-based view of what was found in the uploaded code. After you have already uploaded code and reviewed the initial results in Uploading Code and Reviewing Parsing Results, the next step is to focus on the result details that directly affect your documentation. Look for the parsed symbol list, the symbol type or kind, the source file location, the declaration details, any available comments from the source, and the relationship or dependency views tied to each item.

These result sets help you separate what came directly from the source from what Atloria later turns into generated reference content. The parsed results are the raw extracted facts. Generated reference pages are the reader-facing pages built from those facts. Names, declaration details, source locations, and source comments should stay aligned between both views. If a generated page shows something different, that is a sign you need to review the parsing results before updating documentation.

When you inspect relationships, use the symbol detail view to see how one item connects to another. This is where writers can understand whether a source item belongs under a larger module area, whether it contains child items, or whether it depends on other parts of the codebase. Relationship views are especially useful before drafting a guide, because they show whether a topic stands alone or needs links to related reference pages.

Dependency views add another layer. They help you see which files or packages feed into a source item and which ones rely on it. If one item has many incoming or outgoing connections, it usually needs stronger explanation, clearer related links, or a broader conceptual page.

[SCREENSHOT: Parsing results screen showing symbol list, source file column, comments, and relationship panel]

Mapping symbols and relationships to documentation coverage

To improve documentation coverage in Atloria, start by turning the parsing results into a working inventory. The goal is not to document every parsed item in the same way. Instead, use the parsed list to sort what belongs in public reference content, what belongs only in internal team review, and what appears to be exported but still has no documentation coverage. This gives you a practical way to compare source reality against what readers can actually find in your project.

A simple coverage review usually works best when you group parsed items into categories like these:

Coverage group	What to look for in Atloria	Documentation action
Public reference items	Exported items that appear in generated reference sections	Verify names, details, and links
Internal-only items	Parsed items that should not appear in reader-facing docs	Confirm they stay excluded
Missing coverage	Exported items with no matching page or mention	Add reference coverage or feature guidance

Parent-child relationships help you decide where content belongs. If one parsed item clearly contains several related child items, that often points to a reference page with subsections. If a larger source area connects to several behaviors or workflows, that may belong in a conceptual guide or task-based article instead of a narrow reference page. Relationship views also help you avoid scattering related content across too many pages.

Use dependency links to spot features that span multiple files. A reader may see one public item in a reference page, but the parsing results may show that it relies on shared configuration, supporting types, or other linked source areas. That is often a sign to add cross-links, expand a feature guide, or include a short explanation in a project-level technical page.

As you review, compare parsed exported items against your reference navigation, existing technical pages, and manually written guides. This creates a reliable coverage checklist based on the code already present in Atloria.

Checking generated reference pages against the source material

Generated reference pages in Atloria should reflect the latest parsing results as closely as possible. When you open a reference page, compare what readers see with the parsed source details that feed that page. Focus on the visible fields that matter most to readers: the item name, declaration details, listed inputs or parameters, returned values, inheritance or implementation notes, source comments, and related links. If any of those differ from the parsing results, the page may be outdated or incomplete.

1. Open the generated reference page for the item you want to verify.
2. Open the matching parsed result in the code parsing workspace.
3. Compare the displayed name and declaration details between both views.
4. Check whether listed parameters, return information, and relationship notes still match the parsed result.
5. Review any source comments or descriptive text that Atloria pulled into the page.
6. Confirm that related links point to real, current items that still exist in the parsed results.
7. If you find a mismatch, update the parsing run or documentation workflow before publishing changes.

This check becomes especially important after refactors, renames, or file moves. A generated page may still exist in navigation even though the source item changed location or structure. If the parsing results show the latest declaration but the page still shows an older version, treat the parsed result as the source of truth for validation.

You should also watch for stale relationship details. If a page says an item extends, implements, or links to another source item, confirm that the same relationship still appears in the parsed results. This helps prevent broken reference trails and misleading “See also” sections.

[SCREENSHOT: Side-by-side view of a generated reference page and the matching parsed result]

Using dependency views to strengthen cross-references and architecture docs

Dependency views in Atloria are useful well beyond reference validation. They also help you decide where your documentation needs stronger cross-references and where architecture-focused content is missing. When one source area has many incoming references, that usually means it plays a central role in the project. Those

high-traffic areas often need clearer explanation in technical guides, stronger ownership notes, and better links from surrounding pages.

1. Open the dependency view for a parsed item or source area you are reviewing.
2. Look for items with many incoming references, since these often need stronger conceptual coverage.
3. Review outgoing dependencies to see what supporting items must be understood alongside the main item.
4. Add or improve links between reference pages, setup guides, and architecture pages where those connections matter.
5. Flag dense or circular dependency patterns for documentation follow-up, especially if readers would struggle to understand the flow without extra context.

Outgoing dependency chains are especially helpful when you are writing onboarding or architecture content. They show that a reader may need more than one page to understand how a feature works in practice. If a source area depends on several supporting packages or shared utilities, your documentation should not leave that context hidden inside reference pages alone. Instead, connect the detailed page to a broader explanation that shows setup order, integration points, or shared responsibilities.

Dense dependency clusters can also reveal documentation gaps. If several important items all connect through a shared utility or adapter layer, but that shared layer has little or no explanation in Atloria, readers may struggle to understand the bigger picture. Use those patterns to improve cross-links between technical reference pages, project architecture content, and task-based tutorials.

[SCREENSHOT: Dependency view highlighting a central source area with multiple incoming and outgoing connections]

Building a repeatable review workflow for writers and documentation managers

A one-time parsing review is useful, but Atloria works best when your team turns that review into a repeatable documentation workflow. The most reliable approach is to review parsing results whenever your team reaches a release point, completes a major merge, or refreshes generated reference content. That way, documentation quality does not depend on someone noticing a mismatch by chance.

1. Set a regular review point after major merges, release preparation, or reference regeneration.

2. Rerun parsing for the project workspace at that review point.
3. Compare the latest parsing results with existing reference pages and technical guides.
4. Flag newly added exported items, removed items, and changed declarations for documentation review.
5. Assign follow-up work to the right people based on the issue type.
6. Check publishing criteria before approving updated documentation.

In practice, writers and documentation managers usually focus on different parts of the review. Writers can inspect missing descriptions, weak related links, and gaps between parsed source items and reader-facing pages. Documentation managers can track coverage by project area, module area, or documentation section and decide whether the release is ready to publish.

A simple acceptance checklist inside your team process should include:

- No undocumented public-facing parsed items that are meant to appear in reference content
- No broken source links in generated pages
- No visible declaration mismatches between parsed results and published reference pages
- No missing cross-links for major related items

Store your coverage reports, review notes, and parser snapshots with the rest of your documentation work in Atloria. That history makes it easier to see when a page stopped matching the source and to explain why a change was made.

Fixing common mismatches between parsing results and published docs

When parsing results and published documentation do not match in Atloria, the fix usually starts with identifying which visible part is wrong: the parsed result, the generated page, the relationship links, or the coverage report. Most issues fall into a few repeatable patterns, and each one can be checked from the screens your team already uses for parsing review and documentation publishing.

1. If a generated reference page shows outdated declaration details, rerun parsing for the project and then review the refreshed reference output.
2. If the same item still looks outdated, check whether the source file location in the parsed results matches the current source file you expect.

3. If a documented item is missing from the reference set, review whether it appears in the parsed results at all before changing the documentation page.
4. If related links point to the wrong pages, inspect the current relationship view and confirm the linked items still exist in the latest parsed results.
5. If coverage reports show items that should stay internal, review your team's inclusion and filtering decisions before adding public documentation.

Use this table to guide your review:

What you see	Where to check in Atloria	Likely next action
Outdated declaration details on a reference page	Parsed result and generated page	Rerun parsing and refresh generated content
Missing documented item	Parsed symbol list and coverage view	Confirm the item is included in parsing results
Wrong related links	Relationship view and dependency view	Rebuild or correct linked documentation paths
Internal-only item flagged as missing coverage	Coverage report and reference inclusion rules	Exclude it from public documentation targets

When you work through mismatches this way, you avoid editing the wrong page first. Always confirm what the latest parsing results show before deciding whether the problem is in the source-derived content, the generated reference page, or the coverage rules your team is using.

Overview

This guide focuses on how documentation teams use code parsing results inside Atloria to improve technical documentation quality. It assumes you already know how to upload code and inspect the first round of parsing output, as covered in Uploading Code and Reviewing Parsing Results. Here, the emphasis shifts from "What was parsed?" to "How do we use those results to improve what readers see?"

In Atloria, parsing results are useful for three documentation jobs: checking coverage, validating generated reference pages, and improving cross-links between technical pages. The parsed symbol list helps you see what source items exist. Relationship views help you understand how those items connect. Dependency views help you identify which topics need broader explanation because they sit at the center of a feature or architecture area.

This guide also helps you distinguish between source-derived parsing output and the generated reference pages built from it. That distinction matters because a page can look complete while still being out of date. By comparing the parsed results with published or draft reference content, you can catch stale names, outdated declaration details, missing related links, and gaps in coverage before those issues reach readers.

You will also see how to turn parsing review into a repeatable team workflow. Instead of treating code parsing as a one-off import step, Atloria lets writers and documentation managers use it as an ongoing quality check for technical documentation. The result is more accurate reference content, clearer architecture guidance, and better navigation between related technical topics.

Prerequisites

Before using this workflow in Atloria, make sure you already have access to the project workspace where code parsing results are available. You should also be comfortable opening parsed results, reviewing generated technical pages, and moving between project documentation areas. If you have not done that yet, start with Uploading Code and Reviewing Parsing Results.

You will get the most value from this guide if the following are already in place:

- A project in Atloria with completed code parsing results
- Access to the code parsing workspace for that project
- Existing generated reference pages or technical documentation to compare against parsing results
- Permission to review or update documentation content in the project
- A basic understanding of your team’s documentation structure, such as reference pages, feature guides, and architecture pages

It also helps if your team already has a release or review rhythm. This guide includes a repeatable review process, and that process works best when you can tie parsing checks to regular documentation updates, version work, or release preparation.

If you are reviewing documentation with other team members, agree in advance on which parsed items should appear in public-facing technical content and which ones should stay out of reader navigation. That makes coverage checks much faster and reduces confusion when Atloria shows parsed items that are not meant to become published reference pages.

From here, continue with Uploading Code and Running Parsing Workflows to connect this review process to a broader parsing routine.