ATLORIA

Understanding Parser Coverage and Stack Compatibility

August 25, 2026

Reading parser coverage counts in the repository view

In Atloria, parser coverage is most useful when you read it directly in the **repository analysis** or **code parsing workspace** view, where Atloria shows how much of a repository it can actually understand. Look for the area that summarizes **files scanned**, **supported files**, and any **coverage percentage** shown for the repository. The **files scanned** total tells you how many files Atloria reviewed during analysis. The **supported files** total shows how many of those files match a parser Atloria can use for structure extraction and documentation-related analysis.

[SCREENSHOT: repository analysis view showing files scanned, supported files, and parser coverage percentage]

When Atloria breaks coverage down by language or file type, read those labels as parser families rather than one-extension-per-row. A single count may represent several related file extensions grouped together under one language or parser heading. This helps you understand the real coverage of a codebase without manually checking every extension.

As you review the counts, separate them into three practical groups:

- **Fully parsed files**: files Atloria clearly recognizes and can analyze with strong confidence
- **Partially recognized files**: files Atloria detects, but where extraction may be incomplete
- **Excluded or unsupported files**: files Atloria scanned but could not parse because no matching parser is available

The coverage percentage is the fastest way to judge whether parsing-based features are likely to be useful. A high percentage usually means Atloria can analyze most of the repository's meaningful source files. A lower percentage does not always mean poor fit, though. If the unsupported portion is mostly generated output, archived code, or non-source assets, the repository may still be a strong candidate.

If you need a refresher on where to find language support details before reviewing coverage, see Evaluating Language and Framework Support in Atloria.

Checking which frameworks and stacks Atloria can parse

After you review file coverage, open the **framework** or **stack compatibility** area in Atloria to see which technologies were detected in the repository. This view helps you compare what Atloria found against the files your team expects to be present, such as package manifests, lockfiles, or build-related configuration files that usually signal a specific framework or stack.

[SCREENSHOT: framework compatibility section showing detected technologies and support states]

Use this part of the screen to answer two separate questions:

- Did Atloria **detect the framework or stack** your repository uses?
- Does Atloria show that framework as **supported**, **limited support**, or **unavailable**?

Those states matter because a repository can contain a supported language while still having only partial framework coverage. For example, Atloria may recognize the underlying language files but provide weaker support for framework-specific templates, component formats, or project conventions. In that case, language coverage may look healthy while framework compatibility is only partial.

This is especially important for mixed stacks. A repository may include:

- frontend files with one framework
- backend source in another language
- shared configuration files
- template or component files tied to a specific stack

In Atloria, compare those areas side by side instead of treating the repository as one uniform codebase. Strong support in backend files does not automatically mean the frontend templates are equally covered, and the reverse is also true.

Framework detection is also a good way to confirm whether Atloria is likely to include more than plain source files. If the compatibility area reflects the framework your team uses, that is a good sign that related component files, template files, and configuration files may be included in parsing coverage. If the framework is missing or marked with

limited support, check the file-level counts more carefully before relying on parsing results.

Understanding support indicators before you rely on parsing features

Support indicators in Atloria are the quick signals that tell you how confidently Atloria can work with a language, framework, or file pattern. You may see these as **badges**, **status labels**, or availability markers in the repository analysis view, supported languages area, or parsing workspace. Read them as quality indicators, not just yes-or-no availability.

In practical terms, these statuses usually fall into three working categories:

- **Supported**: Atloria can reliably recognize and analyze the files
- **Limited support**: Atloria can detect some structure, but results may be incomplete
- **Unavailable**: Atloria does not currently parse that file type or framework in a meaningful way

[SCREENSHOT: support badges beside detected languages and frameworks]

These indicators matter because they affect downstream work in Atloria. Lower support can reduce the quality of:

- extracted structure in technical documentation views
- linking between code elements and generated reference material
- completeness of AI-generated documentation based on parsed code
- confidence in repository-wide analysis results

A supported language can still produce gaps if the repository uses uncommon conventions. Watch for situations where the language itself is recognized, but the repository includes:

- **custom file extensions**
- **generated source files**
- **embedded templates**
- **unusual folder layouts**

In those cases, Atloria may show support at the language level while still missing important files in practice. That is why support indicators should always be read together with parser counts and framework detection.

Use the indicator details to decide whether a repository is ready for everyday documentation work or better suited for a smaller evaluation. If the core files your team depends on are marked **supported**, Atloria is usually ready for broader use. If key areas are marked **limited support** or **unavailable**, treat the repository as a pilot candidate until you confirm the output quality in real project work.

Judging fit for monorepos and mixed-language codebases

Monorepos and mixed-language repositories need a more careful review because a single overall coverage percentage can hide major differences between packages, apps, or services. In Atloria, look beyond the top-level parser summary and compare coverage across the parts of the workspace that matter most to your team.

[SCREENSHOT: repository parsing view with multiple packages or services showing different coverage levels]

Start by checking whether parser coverage is spread across the repository or concentrated in only one area. In a monorepo, Atloria may show strong support for one app while another package has much lower coverage. That difference matters if the low-coverage area contains the code your documentation team actually needs.

Pay close attention to whether these areas appear in the totals or seem to be omitted:

- **shared libraries**
- **infrastructure folders**
- **generated directories**
- **archived modules**
- **vendor or dependency folders**

If unsupported counts are coming mostly from generated or third-party content, that may not be a real problem. If the unsupported counts come from shared libraries or business-critical services, the fit is weaker even when the overall percentage looks acceptable.

Mixed-language codebases should be judged by business value, not just by average coverage. Compare high-coverage and low-coverage areas and ask whether Atloria covers the parts that drive your documentation goals. For example, partial stack compatibility may be perfectly acceptable if only one team needs parsing-based documentation for a specific service or app.

This is where Atloria works best as a decision tool. Instead of asking, “Is the whole repository supported?” ask, “Are the important packages, services, and

documentation targets supported well enough?” That approach gives a much more realistic answer for large workspaces with different technologies under one repository.

Deciding when parser coverage is good enough for your team

The right coverage threshold in Atloria depends on what your team wants to do with parsing results. A repository does not need perfect support to be useful, but it does need enough support in the right places. When you review parser coverage, combine three signals instead of relying on one number alone:

- **coverage percentage**
- **number of unsupported files**
- **presence of limited-support frameworks or file types**

A practical decision matrix can help:

- **Strong fit**: core repositories show high supported-file coverage, and the most important source folders are clearly recognized
- **Conditional fit**: overall coverage is acceptable, but critical folders or frameworks show partial support
- **Poor fit**: key source files remain unparsed, or Atloria mainly supports secondary parts of the repository

Different roles may judge the same repository differently. **Project Administrators** often focus on rollout readiness across teams and whether Atloria can support repeatable parsing workflows at scale. **Technical Writers** may care more about whether Atloria can extract enough structure from the code areas used to build documentation pages, reference material, or AI-generated drafts.

[SCREENSHOT: parser coverage summary with supported, limited, and unsupported areas highlighted]

When you make a decision, document the exceptions so the team does not misread the numbers later. Unsupported files may include:

- **vendor code**
- **build artifacts**
- **generated output**
- **archived modules**
- **non-source assets**

Those files can inflate unsupported totals without affecting real documentation work. In Atloria, coverage is “good enough” when the unsupported portion does not block the workflows your team actually plans to use. If the important code is covered and the gaps are mostly low-value files, the repository is usually ready to move forward.

Investigating low coverage and misleading compatibility signals

If parser coverage in Atloria looks lower than expected, do not assume the repository is unsupported right away. First, verify that the repository analysis included the branch, folders, and file types your team intended to review. Low totals can happen when the scan did not include the part of the repository where the main source files live.

Next, compare the compatibility indicators with the actual repository contents. If a framework appears as **unavailable**, check whether the repository includes the usual detection files Atloria would rely on, such as manifest files, lockfiles, or build-related configuration files. If those files are missing, renamed, or not committed, Atloria may not identify the framework correctly even when the source code is present.

[SCREENSHOT: repository analysis screen showing low coverage with framework status details]

When support indicators conflict with what your team expects, inspect patterns that often reduce recognition:

- **custom file extensions**
- **embedded templates inside nonstandard files**
- **generated source**
- **uncommon project layouts**
- **framework files stored outside typical folders**

These cases can make a repository look less compatible than it really is. On the other hand, totals can also look better or worse than they should if the scan includes large excluded areas. Review whether these are affecting the numbers:

- **vendored dependencies**
- **build outputs**
- **generated directories**
- **temporary or archived folders**

A useful habit in Atloria is to compare the top-level compatibility status with the file-level parser counts. If the framework looks supported but the file counts are unexpectedly low, the repository may use conventions Atloria only partially recognizes. If the framework looks unavailable but file-level coverage is still strong, Atloria may be parsing the underlying language even without full framework detection. In either case, use both views together before making a rollout decision.

Overview

This document helps you interpret the parser-related signals Atloria shows when you review a repository in the **code parsing workspace** or related repository analysis views. The goal is not just to see whether a language appears on a supported list, but to decide whether your actual codebase is a good match for parsing-based features such as technical structure extraction, linked reference content, and documentation generation.

The most important areas to review in Atloria are:

- **parser coverage counts**
- **coverage percentages**
- **framework or stack compatibility indicators**
- **support badges showing supported, limited support, or unavailable states**

Taken together, these signals help you answer a practical question: can Atloria meaningfully analyze the parts of the repository your team cares about?

This guide focuses on how to read those signals in real repositories, especially when you are dealing with:

- mixed frontend and backend stacks
- monorepos with multiple packages or services
- repositories that include generated code or vendor content
- frameworks that may be recognized differently from their underlying language files

If you have already reviewed the broader language and framework support lists, this guide builds on that work rather than repeating it. For that earlier step, see Evaluating Language and Framework Support in Atloria.

Use this page when you need to decide whether parsing coverage is strong enough for production documentation workflows, only suitable for a pilot, or too incomplete for the repository you want to analyze. The sections below show how to read the counts,

interpret support indicators, and avoid common mistakes when compatibility signals are misleading.

Prerequisites

Before using this guidance in Atloria, make sure you already have access to a repository or project area where parsing results are visible. You should be able to open the **repository analysis** or **code parsing workspace** view and see parser-related summaries, file counts, or framework detection results. If those areas are not yet available in your project, you may need to complete repository setup first.

This guide assumes you have already done the earlier support review covered in Evaluating Language and Framework Support in Atloria. That earlier document helps you understand what Atloria supports in general. This document is the next step: checking whether your specific repository matches that support well enough to rely on parsing-based features.

It also helps if you know the basic shape of the repository you are reviewing, including:

- the main languages used in the project
- the frameworks your team expects Atloria to detect
- whether the repository is a single app or a monorepo
- which folders contain the business-critical source files
- whether the repository includes generated output, dependencies, or archived modules

You do not need deep technical knowledge to use this guide, but you do need enough familiarity with your project to recognize whether Atloria's counts and compatibility indicators line up with what your team expects to see.

If you are ready to go deeper after this page, continue with Understanding Supported Languages Frameworks and Parser Availability, which looks more closely at how language support, framework recognition, and parser availability relate to each other.