Trace Entity Relationships

September 10, 2026

Trace Entity Relationships

This page explains how to follow the connections between entities in a project's analyzed source. Tracing these relationships helps you understand where a feature begins, what it relies on, and how a change can affect other parts of the product.

Overview

A project is made up of connected entities rather than isolated files or screens. For example, a visible component may render another component, use a hook, depend on a service, or query a model. Following these connections gives you a clearer picture of how a product capability is assembled.

Relationship tracing is especially useful when you need to answer questions such as:

- Which components are involved in a user-facing screen?
- What hooks, contexts, stores, or actions support a component?
- Which service or model is used behind a product feature?
- What may be affected if you update an entity?

Atloria presents relationships as directed connections. The relationship label tells you how the starting entity connects to the next entity. For example, a chain can move from a **COMPONENT** to a **HOOK**, then to a **PAGE**, **LAYOUT**, **ROUTE**, **CONTEXT**, **PROVIDER**, **STORE**, **CONTROLLER**, **SERVICE**, and **REPOSITORY**. This allows you to trace a feature from its visible interface toward the supporting product logic.

> 📷 *Screenshot pending: A focused project view showing the Source area used to begin source entity discovery.*

Prerequisites

Before you begin, make sure that:

- You have access to the relevant project in **Projects**.
- The project source has been added and is available for analysis.
- You know the entity, feature, or product area you want to investigate, such as a component name, service name, or screen behavior.
- You can open the project's focused view by selecting **Open Project Dashboard**.
- You understand whether you are tracing from a user-facing entity toward dependencies, or from a supporting entity toward the features that use it.

Step-by-Step Instructions

1. Open the project you want to investigate.

   Start from **Projects**. If you are working across multiple projects, use **Search projects...** to locate the project. You can also use **Search across all projects...** when you need to find a project or content beyond the current project list.

   Select the project and choose **Open Project Dashboard** to enter the focused project view.

2. Go to the project's source information.

   In the focused project area, select **Source**. This is the starting point for examining the entities identified from your project source and understanding how they connect.

   > 📷 *Screenshot pending: The Source view with the "Search entities…" field available for locating a project entity.*

3. Find the entity you want to trace.

   Use the **Search entities…** field to search for a known entity. Enter a specific name when possible—for example, the name of a component, hook, context, service, repository, or model.

   Begin with the entity closest to the question you are trying to answer. If you are investigating a visible product behavior, start with the related component or page. If you are investigating data access or system behavior, start with the service, repository, or model involved.

4. Review the entity's relationships.

Open the entity and review the relationship labels shown with its connected entities. Treat each label as a statement about the connection:

- **RENDERS** indicates that one component displays another component.
- **USES HOOK** indicates that an entity relies on a hook.
- **USES CONTEXT** and **PROVIDES CONTEXT** show how shared context is consumed or made available.
- **USES STORE** and **DISPATCHES ACTION** show interactions with application state.
- **CALLS API** and **IMPLEMENTS API** distinguish a caller from the entity that provides the API behavior.
- **DEPENDS ON** identifies a direct dependency.
- **USES SERVICE**, **INJECTS**, and **QUERIES MODEL** help you follow a feature into its supporting service and data layers.

5. Follow the relationship in the direction shown.

   Select a connected entity to continue the trace. Read the relationship label before moving on so that you preserve the meaning of the chain. For example, a path may show that a component **USES HOOK**, the hook **USES CONTEXT**, and the context **PROVIDES CONTEXT** to the rest of the feature.

   Continue until you reach the level needed for your investigation. A user-interface question may only require tracing through components and hooks. An impact-analysis question may require continuing through stores, controllers, services, repositories, and models.

6. Trace backward when you need to understand impact.

   To understand what a change could affect, begin with the entity you plan to change and look for entities that render it, use it, depend on it, or call it. This reverse investigation is useful before changing shared components, contexts, services, or models.

7. Confirm the source details when necessary.

   Select **View Source** when you need to verify the implementation context for an entity or relationship. Use this to confirm names and surrounding source details after you have identified the relevant relationship path.

   If you need generated explanatory material for a project area, select **User Docs** where available and review the applicable user-facing documentation alongside

your relationship trace.

> 📷 *Screenshot pending: An entity detail view showing relationship labels such as RENDERS, USES HOOK, USES SERVICE, and QUERIES MODEL.*

Tips and Best Practices

Start narrow. Search for one known entity rather than trying to understand the entire project at once. A specific component, hook, or service provides a practical anchor for the trace.

Read relationship labels literally and in order. **USES SERVICE** does not mean the service renders the component, and **CALLS API** does not mean the entity implements that API. The direction and wording of the relationship are important when you are determining ownership and dependency.

Use a layered approach. For user-interface investigation, trace from **COMPONENT** through related components, hooks, contexts, and stores. For deeper behavior analysis, continue from controller and service relationships toward repositories and models. This keeps the investigation focused while still allowing you to follow the full dependency chain when needed.

Be cautious with shared entities. A context, provider, store, service, or model can support multiple features. Before changing one, trace both its dependencies and the entities that use it to identify possible product-wide effects.

Related Pages

- Manage Projects
- Explore Project Source
- Review Project Changes
- Compare Project Versions
- Use Technical Documentation

Troubleshooting

I cannot find the entity I need

Use **Search entities…** with the entity's exact name where possible. If you are searching from a business-feature name rather than a source entity name, start with

the visible component or page associated with that feature and trace from there.

The relationship chain becomes too large to follow

Stop at the layer relevant to your question. For example, stop after hooks and context for a screen-behavior question, or continue through services and models only when you need to understand backend or data dependencies. Trace one branch at a time and record the labels between entities.

I am unsure whether an entity is safe to change

Trace backward to identify entities that **RENDER**, **USE**, **DEPEND ON**, or **CALL** the entity. Then use **View Source** to validate the context before making a change. Shared contexts, stores, services, and models should be treated as potentially high-impact entities.