

Establishing Provenance: Giving Information a Traceable History

A practical introduction to PROV-O, knowledge graphs, and trustworthy AI

Michael Iantosca

Senior Director of Content and Knowledge Engineering

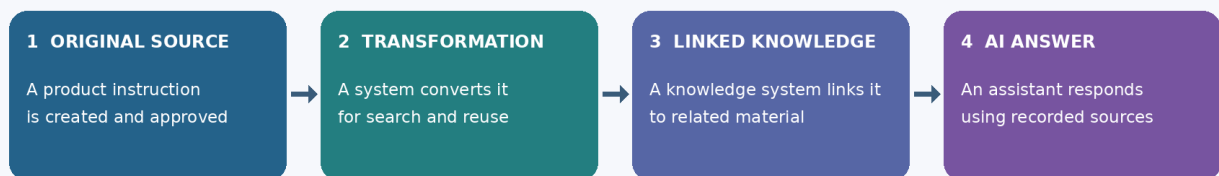
Avalara

When an AI assistant answers a software question, it should be possible to discover where the information came from, which version it describes, whether it changed, and which sources the system actually used. That traceable history is called provenance. Provenance provides evidence of origin and processing, but does not by itself prove that an answer is correct.

Good provenance is not the same as simply adding a source link to a document. It connects information to the people, systems, and activities that created or changed it. That becomes increasingly important when knowledge moves among documentation, support systems, search indexes, and AI assistants.

HOW INFORMATION GETS ITS HISTORY

A simple example of PROV-O at work



PROV-O records the connections:

WHAT was used • WHO or WHAT did the work • HOW an output was created

This is traceability, not automatic proof that the content is correct.

What is PROV-O?

PROV-O is a standard vocabulary from the World Wide Web Consortium (W3C) for describing the history of information. It is built around three straightforward ideas: an entity (a document, fact, or answer), an activity (a review, conversion, or search), and an agent (a person or software system involved). By connecting these, organizations can record how one piece of information led to another.

The business need and imperative for technical documentation

Software technical documentation is no longer simply a published reference for people. It is increasingly a knowledge source for support organizations, customer-facing AI assistants, automated workflows, and software agents. Whether knowledge begins in DITA, Markdown, AsciiDoc, reStructuredText (RST), or another format, organizations need to know which source version was used, what processing occurred, and who or what was responsible as content moves into digital twins, search indexes, support systems, and generated answers.

The business imperative is traceability and accountability at enterprise scale - not merely better citations. As agentic workflows automate source-code analysis, documentation-gap detection, drafting, SME review routing, transformation, and publication, a machine-readable record becomes more important, not less. Without it, investigating incorrect guidance, reconciling conflicting versions, and demonstrating that review controls were performed can become increasingly difficult.

Six practical business outcomes

- **Reduce the risk of incorrect guidance.** Trace an instruction to a precise source version and identify prior or superseded information; factual correctness still requires separate validation.
- **Make AI answers accountable.** Follow recorded retrieval and transformation events back to the evidence actually supplied to an AI system, rather than relying solely on a linked Knowledge Center page.
- **Accelerate documentation maintenance. Support impact analysis for changes to source files and topics, published content, graph resources, and derived search artifacts where dependencies have been captured.**
- **Align documentation with software releases.** Relate technical instructions to documented software versions, configurations, and release evidence using enterprise applicability models alongside provenance.
- **Reduce support investigation effort.** Shorten the work of finding the origin of outdated or contradictory guidance and determining where corrective action is needed.
- **Strengthen governance and auditability.** Preserve evidence of transformation, review participation, agent actions, publication, and separately recorded approval decisions.

Why PROV-O in an existing digital-twin architecture?

Provenance is the business requirement; PROV-O is a standards-based way to implement it. A database or execution log can also record history, but the W3C PROV-O vocabulary provides shared RDF semantics for entities, activities, agents, and derivations across linked digital twins. It complements existing domain ontologies, iIRDS where used, source formats such as DITA/XML, Markdown, and AsciiDoc, and the GraphDB infrastructure. Software applicability, approval status, and correctness remain governed by the appropriate domain models and controls.

How to demonstrate value

Pilot the approach and compare: (1) time to trace a disputed AI answer to the exact source versions and processing records; (2) time to identify downstream artifacts affected by a documentation change; and (3) the percentage of published content with complete, verifiable source, transformation, and approval evidence. These measures establish value without assuming that provenance alone guarantees accuracy or cost savings.

The central business objective is to make enterprise technical knowledge not only accessible and reusable, but traceable, maintainable, and accountable throughout its lifecycle.

Why use a knowledge graph?

Provenance does not require a knowledge graph. A well-designed database can also record it. A knowledge graph is particularly useful when information crosses many systems and formats. It can connect a software API

instruction to its exact file revision, reviewer, transformation run, related support case, software release, and AI answers that used it. Those relationships can then be queried across a unified semantic layer.

A familiar example

Imagine a software platform changes an authentication API. In version 4.2, clients must send a new request header when refreshing access tokens. The instruction appears in a DITA API topic, a Markdown developer guide, an AsciiDoc integration tutorial, a support article, and an AI assistant response. Keeping the new instruction consistent matters. Provenance goes further: it records which engineering change introduced the requirement, which source versions were reviewed and updated, what was republished or reindexed, and which AI answers used earlier guidance. The provenance trail helps investigators locate affected content; technical validation still determines whether the guidance is correct.

Putting it into practice

- **Start with one important fact.** Choose a specific instruction or claim used in more than one place.
- **Give every version an identity.** Record the exact source version rather than only a general page address.
- **Record the important events.** Capture approvals, conversions, updates, and the systems or people involved.
- **Connect related systems.** Link documentation, support records, and search or AI outputs where the relationship is known.
- **Trace an actual AI answer.** Verify that its recorded evidence leads to the exact source versions supplied to the model.
- **Check quality separately.** Provenance shows the history of an answer; subject-matter review and testing help establish whether it is correct.

A sensible starting point

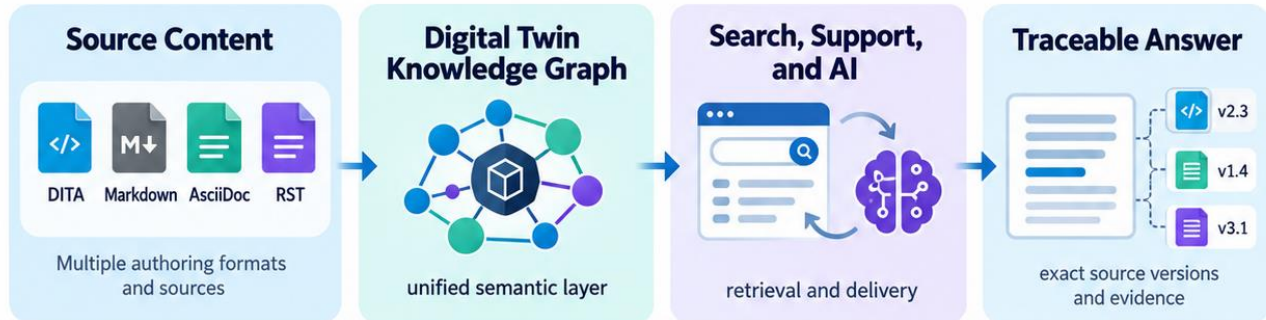
Pilot the approach with one software API instruction documented in one or more formats, one related support record, and one AI-generated answer. Show that an investigator can work backward from the answer to precise source versions and processing steps. Once the trail is reliable, extend the approach across other systems and formats.

The central idea: Trustworthy AI should not only provide an answer. It should preserve enough evidence to explain where that answer came from and how the information reached it.

Why Provenance Matters in Technical Documentation

Using PROV-O with a Digital Twin Knowledge Graph

A practical model for traceable, accountable software knowledge



PROV-O records

Captures the provenance of content, transformations, and usage.



Entity

What (e.g., a topic, file, or dataset)



Activity

What happened (e.g., created, transformed, published)



Agent

Who or what (e.g., author, tool, or system)

Business outcomes



Reduce incorrect guidance



Make AI answers accountable



Speed documentation maintenance



Align docs with software releases



Reduce support investigation effort



Strengthen governance and auditability



DITA + iIRDS shown as a reference implementation.
The approach also supports mixed content formats.

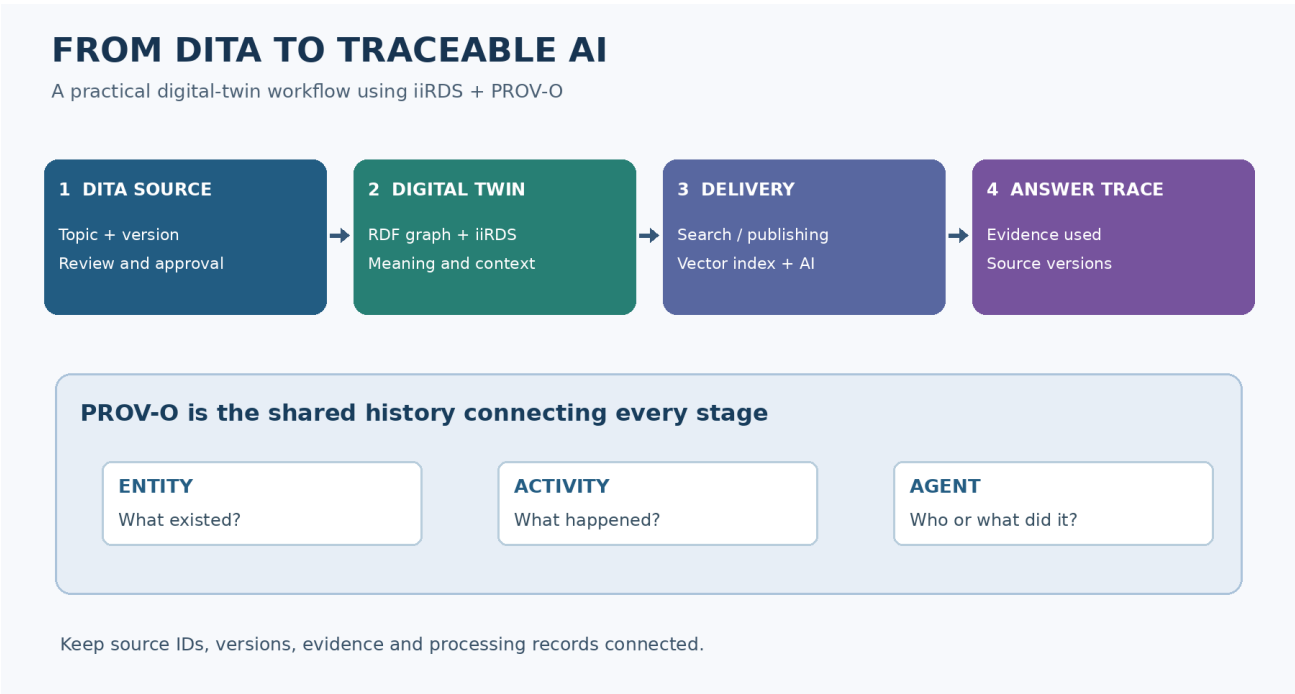


Trustworthy AI needs traceable knowledge, not just linked pages.

Part 2: Implementing Provenance in a Multi-Format Digital Twin

A practical implementation using DITA and iiRDS, with a model applicable to Markdown, AsciiDoc, RST, and mixed-source knowledge

Part 1 introduced provenance as the traceable history of information. This part explains one practical way to implement it in a knowledge-graph digital twin. DITA and iiRDS provide the detailed reference implementation, not a prerequisite for adoption. Sources can also include Markdown, AsciiDoc, reStructuredText (RST), API specifications, and other structured or unstructured content. Their representations can coexist in a unified semantic knowledge graph when ingestion preserves source identity, version, structure, applicability, and relationships. PROV-O supplies a common, format-independent record of the entities used, activities performed, and people or software agents responsible. The degree of structural detail available depends on each source and its ingestion pipeline.



Distinct roles for source formats, metadata, the digital twin, and PROV-O

Standard / system	Its job, in plain language
DITA (reference source)	Structured authoring source with topics, maps, reuse, and revisions.
iiRDS	Standardized classification and context metadata, illustrated here with DITA; usable with other source formats where mapped.
RDF / digital twin	Unified, machine-readable representation of content from multiple formats, with shared concepts and relationships.

PROV-O

The history: source entities, processing activities, and responsible people or systems.

Markdown / AsciiDoc / RST

Additional source formats whose versions, structure, links, and metadata can be mapped to the same graph.

PROV-O complements rather than replaces source formats, iiRDS, or the enterprise ontology. In the DITA reference implementation, iiRDS provides classification and applicability metadata. Other sources may use equivalent metadata mappings into the enterprise ontology and may also use iiRDS where appropriate. An RDF store such as GraphDB can hold the digital-twin relationships and provenance together; detailed execution traces can remain in a dedicated trace store.

A six-step implementation plan

- 1. Identify exact source versions** Assign stable identities to each source artifact and revision, whether a DITA topic/map, Markdown file, AsciiDoc page, or RST document. Preserve repository location, commit or version, publication scope, timestamp, and ideally a content hash. A changing web URL is not sufficient.
- 2. Create or update the digital twin** Transform content into RDF resources and relationships while preserving each source's available structure, links, reuse or inclusion context, release applicability, and exact version. For DITA, retain topic and map relationships and attach native iiRDS classifications where applicable. For Markdown, AsciiDoc, or RST, map available headings, attributes, links, and repository metadata into the shared semantic model without pretending that all formats carry identical structure.
- 3. Record transformations** Represent each ingestion, conversion, indexing, or publishing run as a PROV-O activity. Record which source versions it used, which RDF or search artifacts it produced, when it ran, and which software or person performed it.
- 4. Capture review and release decisions** Represent SME review, editorial approval, and release as separate activities, regardless of authoring format. Record participating agents, input versions, and decisions. Approval status requires a separate business property or evidence record; PROV-O participation does not itself mean approved.
- 5. Carry identifiers into search and AI** Preserve the source artifact/version ID, graph version, and processing-run ID in published pages, vector entries, and AI evidence packages. Record which evidence was actually supplied to an answer, with query/template and model versions. Apply the same contract across DITA, Markdown, AsciiDoc, and other sources.
- 6. Validate and test the trail** Use SHACL to check required provenance identifiers and links. Trace a sample AI answer backward through retrieval, digital-twin representation, ingestion, review, and the exact source revision. Test factual accuracy separately.

A worked software example: updating an authentication API

A platform release changes token-refresh requests so clients must include a new header. Engineering records the change for software version 4.2. A DITA API topic is updated and approved; a Markdown developer guide and an AsciiDoc integration tutorial are also revised. The digital twin links the corresponding instructions to the API and applicable release. iIRDS classifies the DITA content in this reference implementation, while comparable metadata mappings identify the other sources. PROV-O records the reviews, conversions, and indexing activities for each exact source revision. When an AI answer is challenged, investigators can trace the evidence it received to the source versions and processing history, then check the answer against the authoritative API behavior.

A small example of the provenance record

This illustrative Turtle snippet shows a DITA-based path for the authentication API update. The same PROV-O relationships apply to Markdown and AsciiDoc source revisions. The example.org names are placeholders rather than official vocabulary terms.

```
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix ex: <https://example.org/knowledge/> .

ex:auth-api-dita-v4 a prov:Entity, ex:DITATopicVersion ;
  prov:wasRevisionOf ex:auth-api-dita-v3 .

ex:auth-api-markdown-v6 a prov:Entity, ex:MarkdownRevision .
ex:auth-api-asciidoc-v2 a prov:Entity, ex:AsciiDocRevision .

ex:ingestion-run-42 a prov:Activity ;
  prov:used ex:auth-api-dita-v4, ex:auth-api-markdown-v6, ex:auth-api-asciidoc-v2 ;
  prov:wasAssociatedWith ex:content-graph-service .

ex:auth-api-rdf-v4 a prov:Entity ;
  prov:wasDerivedFrom ex:auth-api-dita-v4 ;
  prov:wasGeneratedBy ex:ingestion-run-42 .

ex:content-graph-service a prov:SoftwareAgent .
```

The implementation must separately record software-release applicability, iIRDS or equivalent classifications, review decisions and approval evidence, index/graph versions, and which sources were supplied to an AI answer. Shared subject matter across formats does not imply the files are identical, equally authoritative, or interchangeable.

What to include in the first pilot

- **Source:** One API instruction in DITA and, where available, a related Markdown or AsciiDoc revision; identify exact source versions.
- **Twin:** RDF representations of the sources, shared API/release concepts, and iIRDS classification for DITA where applicable.
- **History:** Review and ingestion activities for each source, plus one published or indexed output.
- **AI:** One question, its retrieved evidence, and one recorded answer.
- **Verification:** A query that traces the answer to the exact source version; a SHACL check for missing provenance records.

Expansion: mixed formats and linked digital twins

Once the documentation trail works, extend it to additional source formats and linked twins for support cases, product requirements, and software releases. Each format and system retains its native semantics and version

PROV-O and enterprise knowledge provenance • General-audience briefing

history while shared identifiers and provenance relationships connect the derived knowledge. PROV-O records how information was ingested, combined, evaluated, and used. A support case may motivate a documentation change without becoming an authoritative API specification.

Guardrails

Keep three checks separate: provenance tells where an item came from and what happened to it; conformance checks whether its structure meets defined rules; correctness requires evidence and the appropriate expert judgment. Also distinguish “retrieved for an AI answer” from “actually supports this specific claim.” Claim-level support needs its own evaluation.

Implementation outcome: an organization can move from “here is the page the AI linked” to “here is the exact approved source version, the transformations it passed through, and the evidence supplied for this answer.”

Standards: W3C PROV-O (<https://www.w3.org/TR/prov-o/>); iiRDS (<https://iirds.org/>); OASIS DITA (<https://www.oasis-open.org/committees/dita/>).