Reliable Context

RELIABLE CONTEXT / RESEARCH & PRACTICAL GUIDANCE

Safe ontology releases beyond SHACL conformance

> Reliable Context is a kernel of truth: a system of record for organizations that fuse private and public information and must defend every value they act on. One canonical record per real thing, a cited source and an observation date behind every value, unknowns declared rather than left blank, and any past state of the record reproducible.

Published by Reliable Context · 2026-09-22 · ontology release beyond shacl conformance · ontology versioning · shacl validation · ontology governance · knowledge graph governance · ontology change impact · owl 2 · sparql regression testing · ontology provenance

Original article: https://reliablecontext.com/articles/safe-ontology-release-beyond-shacl



In this article

1. Executive Summary
2. Introduction and Background
3. Key Changes in Release Assurance
4. Change Classification and Compatibility
5. Implementation Considerations and Process Changes
6. Data Analysis and Evidence
7. Release Evidence Package and Rollback
8. Implications and Future Directions
9. Frequently Asked Questions (FAQs)
10. Conclusion

Executive Summary

A green Shapes Constraint Language (SHACL) report proves a narrow proposition: for a particular data graph, shapes graph, processor configuration, and conformance-severity policy, the processor returned no disallowed results. It does not prove that an ontology is logically consistent, that imports are stable, that identifiers retain their meaning, that saved queries preserve their answers, or that consumers can migrate. SHACL 1.2 makes the validation context easier to record, but it remains a **W3C Working Draft dated 18 September 2026**, not a Recommendation (w3.org). The draft itself says that passing all suite tests does not establish complete conformance (w3.org). Ontology evolution is broader by construction: a peer-reviewed survey describes it as a multistage process drawing on several fields (doi.org).

The safe release decision therefore needs **nine independent gates**: artifact integrity, SHACL fixtures, logical consistency, entailment change, identifier stability, import closure, queries and competency questions, consumer contracts, and migration and rollback. A gate may be pass, fail, or unknown. Unknown is not green. Compatibility is also consumer-relative. Research on safe ontology replacement states that its meaning depends on the intended application, while query-serving systems require stable answers to relevant ontology-mediated queries (arxiv.org).

The operational outcome is one of three statuses. **Publish as compatible** only when every applicable gate passes and measured consumer behavior remains inside an explicit tolerance. **Publish as breaking with migration** when the semantic or behavioral delta is intentional, versioned, documented, and supported by executable mappings or dual-run evidence. **Stop the release** when an applicable gate fails, an import or test is unavailable, a stable identifier is reused, or a material outcome remains unknown. OWL 2 defines imports transitively and advises against mixing versions from one ontology series (w3.org). OBO Foundry adds a practical requirement: official release files must remain unchanged and retrievable through unique version IRIs (obofoundry.org).

A worked fixture in this report contains **12 input triples, 4 expected query rows, 5 expected entailments, and zero baseline SHACL results**. The proposed release produces 3 query rows, 4 entailments, and 2 new SHACL results. That is a **25% row loss**, **20% entailment loss**, and an unambiguous stop until migration evidence explains the deltas. The evidence packet should retain both artifacts, their checksums, import

manifests, configurations, results, approvals, and rollback target. DCAT defines a checksum as a way to verify that a file or package has not changed (w3.org). RO-Crate guidance separately recommends recording the actual tool version that produced a file (researchobject.org).

Introduction and Background

An **ontology release** is a published, immutable version of an ontology artifact together with the imports and operational evidence needed to interpret it. A **shapes graph** is the Resource Description Framework (RDF) graph containing SHACL shapes. A **data graph** is the RDF graph being validated. An **import closure** is the release plus every ontology reached through direct or transitive imports. A **consumer** is any query, transformation, reasoner, interface, policy, or downstream dataset whose behavior depends on the released meaning. **Backward compatibility** means that a defined set of consumer-observable obligations still holds. It is not a property conferred by a version label.

These definitions separate four objects that teams often collapse: syntax, constraints, logic, and behavior. SHACL concerns RDF graph structure; OWL 2 consistency asks whether a model exists (w3.org). Query compatibility asks whether materialized or virtual answers change. Identifier compatibility asks whether an Internationalized Resource Identifier (IRI) still denotes the same intended thing. Noy and Klein identify several distinct compatibility dimensions and explain why explicit semantics make ontology evolution unlike ordinary database-schema evolution (research.vu.nl).

The governing principle is therefore **bounded evidence**. A result should state what artifact, graph, closure, engine, configuration, fixture, and policy it covers. NIST's ontology evaluation framework distinguishes intelligibility, fidelity, craftsmanship, fitness, and deployability, which is a useful warning against reducing quality to one report (nvlpubs.nist.gov). Reliable Context's relevant first-party perspective is similarly about evidence boundaries, not ontology execution: it describes “a cited, verbatim, dated source behind every value” and replay of past record states. For release governance, the analogous requirement is that every gate decision remain traceable to the exact inputs and observations used.

Key Changes in Release Assurance

From a green report to a bounded conformance claim

SHACL validation is a mapping from inputs to validation results (w3.org). The boolean `sh:conforms` is meaningful only inside that mapping. Conformance depends on the processor's severity and execution settings, while processors may truncate returned details. Result counts are therefore comparable only when configurations are identical and truncation is disabled or recorded.

A release gate should record at least:

- **Data graph:** content hash, graph IRI, version IRI, and fixture identifier.
- **Shapes graph:** content hash, version IRI, import closure, and active shape set.
- **Processor:** product, exact version, execution profile, and limits.
- **Severity policy:** disallowed severities and any explicitly tolerated results.
- **Outcome:** conformance boolean, complete result graph, runtime status, and evidence URI.

- **Coverage:** targeted classes, focus-node selection, disabled shapes, and known exclusions.

SHACL 1.2 helps by allowing a report to identify the data graph, shapes graph, and processor configuration so that it can be self-contained with respect to context (w3.org). A portable gate should compare machine fields such as focus node, path, source shape, constraint component, severity, and value, not human message strings. OpenRiC's draft offers one implementation example: it requires its SHACL check to run in continuous integration (openric.org).

From file comparison to logical and behavioral difference

A text or triple diff answers what serialization changed. It does not answer what follows logically. ROBOT's ontology diff deliberately compares axioms while ignoring serialization differences, and it accepts separate catalogs when two versions resolve different imports (robot.obolibrary.org). A separate inference diff uses a reasoner and reports changed inferred subclass axioms (obophenotype.github.io). Both are needed because an apparently small axiom edit can alter many classifications, while a large annotation edit may alter none.

The release record should distinguish:

- **Structural delta:** added and removed declarations, axioms, shapes, annotations, and imports.
- **Logical delta:** added and removed entailments under the declared semantics and reasoner.
- **Validation delta:** newly failing, newly passing, and unchanged SHACL result identities.
- **Query delta:** added and removed rows or graphs for an approved query suite.
- **Consumer delta:** changes observed by APIs, transformations, forms, rules, and exports.
- **Migration delta:** records transformed, unmapped values, collisions, and exceptions.

Logical-difference research formalizes ontology difference through queries whose answers change and distinguishes subsumption, instance, and conjunctive-query differences (arxiv.org). This supports a practical rule: compatibility must be defined in the same observation language that the consumer uses.

From mutable locations to versioned identity and closure

OWL 2 permits an ontology IRI and an additional version IRI that identifies a particular version (w3.org). The ontology IRI can resolve to the current release, while each immutable artifact remains retrievable at its version IRI. OBO Foundry operationalizes this by requiring the version string in the artifact and version IRI to match, and versioned persistent URLs to keep resolving (obofoundry.org).

Import closure must be locked with the same rigor. A current import IRI that resolves differently tomorrow makes a nominally identical release behave differently. Record each resolved document IRI, declared ontology IRI, version IRI, digest, media type, and retrieval date. Treat an unavailable import as a failed or unknown gate, not as an empty module. OBO Academy describes a typical release in which modules and imports are combined, and its release guidance calls for a stable workflow (oboacademy.github.io).

Change Classification and Compatibility

No edit type has a universal semantic-version label. A label change can break lexical search; an additive class can alter closed-world exports; a stricter cardinality can invalidate existing data; and deleting an apparently unused IRI can break an external query. **Semantic versioning is a local communication policy**, not a proof of logical or consumer compatibility. The draft OpenRiC governance profile makes that locality explicit by defining its own major, minor, and patch meanings, and it remains a draft open for comment (openric.org).

Table 1 classifies frequent edits by possible breakage surface, required evidence, default version action, and migration artifact.

Change class	Example axiom or shape change	Possible breakage surfaces	Required tests	Default version action	Migration artifact
Documentation only	Correct `rdfs:comment`; preserve language tags	Search, display, generated documentation	Serialization lint, language coverage, UI snapshots	Patch if local policy permits	Change note
Additive term	Add class, property, or non-targeting shape	Classification, enumerations, exports, closed-world consumers	Reasoner, query suite, enumeration contract tests	Minor only if all obligations pass	Adoption note and new-term map
Domain or range	Add `rdfs:domain` or `rdfs:range`	New type entailments, inconsistency, query cardinality	Consistency, entailment diff, saved queries	Breaking if observed answers or classifications change	Before-and-after entailment map
Cardinality or SHACL constraint	Raise `sh:minCount` from 0 to 1	Existing data conformance, forms, loaders	Complete fixture validation, production-like samples, form tests	Breaking when existing accepted data fails	Repair query and exception policy
Disjointness	Add `owl:disjointWith`	Consistency and classification	Full-closure reasoner and unsatisfiable-class report	Breaking unless affected signature is provably unused	Affected-individual report
Identifier replacement	Replace `ex:oldTerm` with `ex:newTerm`	Queries, stored data, links, code, caches	IRI inventory, dependency scan, dual-query tests	Breaking	Explicit mapping, deprecation, sunset date
Deletion	Remove class, property, axiom, or shape	Every consumer surface	Usage scan, query and contract suite, closure diff	Breaking by default	Tombstone, replacement guidance, data rewrite
Import change	Move from dependency v2 to v3	Parseability, entailments, shape targets, licenses	Resolved manifest, consistency, entailment and query diffs	Determined by observed closure delta	Locked manifest and rollback bundle

The table is intentionally conservative. OBO's term-stability policy says that a substantial referent change requires a new term and IRI, and deprecated terms should carry handling guidance (obofoundry.org). The release owner may choose less conservative labels only after defining the consumer set, observations, and permitted tolerances.

Implementation Considerations and Process Changes

Execute the gates in dependency order

Cheap, deterministic failures should stop expensive downstream work. A practical pipeline runs as follows:

1. **Freeze candidate inputs.** Hash ontology, shapes, fixtures, queries, mappings, catalogs, and configuration.
2. **Parse and lint.** Confirm media types, namespace declarations, duplicate ontology headers, IRI policy, and shapes well-formedness.
3. **Resolve imports.** Materialize and hash the complete closure; reject incompatible versions from one series.
4. **Classify and reason.** Check ontology consistency, unsatisfiable named classes, and declared profile compliance.
5. **Run SHACL fixtures.** Execute positive, negative, boundary, and migration fixtures with result limits disabled.
6. **Compare entailments.** Diff an approved signature rather than every theoretically expressible consequence.
7. **Run competency questions.** Execute formalized questions and verify expected bindings or ASK booleans.
8. **Compare saved queries.** Normalize result sets, preserve multiplicity when relevant, and canonicalize blank nodes.
9. **Run consumer contracts.** Test generated code, interfaces, transforms, exports, and policy rules.
10. **Exercise migration and rollback.** Measure coverage, exceptions, reversibility, and restoration time.
11. **Assign status.** Compatible, breaking with migration, or stopped, with an accountable approver.
12. **Publish atomically.** Release immutable artifacts, manifests, notes, evidence, and resolver updates together.

Shapes well-formedness deserves an explicit preflight because SHACL engines are not strictly required to check it (w3.org). Engine-specific behavior is a portability boundary. A project that relies on it should name and test its engine as part of the consumer contract. The Ontology Development Kit similarly supports release diffs and broader SPARQL-based checks across ontology components (github.com).

Represent pass, fail, and unknown separately

Binary dashboards hide missing evidence. A release matrix should distinguish **pass**, **fail**, and **unknown**, while detailed records retain whether unknown means indeterminate, inapplicable, or untested. OpenRiC's draft also requires a public, diffable change log, illustrating how governance evidence can remain inspectable (openric.org).

Table 2 shows the minimum release-gate matrix. Evidence URIs are illustrative identifiers and must resolve inside the release evidence store.

Gate	Pass condition	Fail condition	Unknown condition	Evidence URI	Decision effect
Artifact integrity	All expected files parse and hashes match	Parse, lint, or hash failure	Candidate or manifest absent	`urn:release:42:integrity`	Stop on fail or unknown
SHACL fixtures	Expected result identities and conformance match	New unexplained result or missing expected result	Engine failure, truncation, or incomplete fixture	`urn:release:42:shacl`	Stop pending disposition
Logical consistency	A model exists and no prohibited unsatisfiable class appears	Inconsistency or prohibited class	Reasoner timeout or unsupported construct	`urn:release:42:reasoner`	Stop
Entailment change	Delta is empty or approved within signature	Unapproved added or removed entailment	Signature or reasoner missing	`urn:release:42:entailments`	Breaking or stop
Identifiers	Published IRIs retain identity or are deprecated	IRI reused or meaning overwritten	External usage not assessed	`urn:release:42:iris`	Stop
Import closure	Every pinned dependency resolves and hashes match	Mixed versions or digest mismatch	Import unavailable	`urn:release:42:imports`	Stop
Queries and questions	Expected bindings, multiplicities, and booleans hold	Result delta outside tolerance	Nondeterministic or missing oracle	`urn:release:42:queries`	Breaking or stop
Consumer contracts	Named consumers pass agreed tests	Contract regression	Consumer not tested	`urn:release:42:contracts`	Stop or scoped breaking release
Migration and rollback	Coverage and reversibility thresholds pass	Unmapped values or failed restore	No rehearsal	`urn:release:42:migration`	Stop

The matrix makes uncertainty actionable. An unavailable import is not a pass because the previous cache happened to work. A reasoner timeout is not evidence of consistency. A consumer omitted from testing is not compatible by default. NIST recommends protecting existing deployed functionality and turning successful competency questions into regression tests (nvlpubs.nist.gov).

Normalize query evidence

Saved SPARQL Protocol and RDF Query Language (SPARQL) outputs need semantic normalization. SPARQL results are multisets, so duplicates may be meaningful (w3.org). Result order must be declared, and generated blank-node labels cannot safely serve as cross-release keys.

Accordingly, the comparison policy should specify:

- **SELECT:** variables, datatype and language normalization, multiplicity, and ordering policy.
- **ASK:** exact boolean expectation and active dataset.
- **CONSTRUCT or DESCRIBE:** RDF graph isomorphism or canonicalization method.
- **Entailment regime:** simple, RDF Schema, OWL profile, custom rules, or none.
- **Dataset:** default and named graphs, including fixture version identifiers.
- **Tolerance:** exact equality or a documented set of allowed added and removed answers.

Data Analysis and Evidence

Synthetic release calculation (Hypothetical Example)

This section provides a reproducible synthetic calculation, not an industry benchmark. The purpose is to show how a release that is superficially green can still be incompatible. The fixture uses a compact data graph of **12 asserted triples**, a baseline shapes graph, a candidate shapes graph, **3 saved queries**, **2 competency-question ASK tests**, and an approved entailment signature containing **5 facts**.

In the baseline:

- **SHACL:** 0 result identities and `sh:conforms true`.
- **Query Q1:** 3 rows listing active products.
- **Query Q2:** 1 row listing an organization inferred from a domain axiom.
- **Query Q3:** ASK returns true for a legacy identifier mapping.
- **Competency questions:** 2 of 2 expected booleans pass.
- **Entailments:** 5 of 5 approved facts are present.

The candidate adds a minimum-cardinality constraint, narrows a property domain, deletes the legacy mapping, and updates one import. It still reports `sh:conforms true` on the candidate-only clean fixture, but on the frozen baseline fixture it produces **2 new result identities**. The saved suite returns **3 rows instead of 4**, the approved signature retains **4 of 5 entailments**, and one competency question changes from true to false.

The measured deltas are:

- **Query row loss:** `(4 - 3) / 4 = 25%`.
- **Entailment loss:** `(5 - 4) / 5 = 20%`.
- **Competency-question regression:** `(2 - 1) / 2 = 50%`.
- **New SHACL results:** `2 - 0 = 2`.
- **Candidate-only false reassurance:** 0 reported results because the changed fixture excludes legacy data.

These calculations support **stop**, not “publish compatible.” If the changes are intentional, the owner can convert the outcome to **publish as breaking with migration** by supplying a legacy-identifier mapping, data repair for the cardinality constraint, an explanation of the removed entailment, updated query expectations, and passing dual-run consumer tests. Competency questions are legitimate functional requirements, but empirical work compared observed question patterns with **46 SPARQL-OWL query signatures** and released a set of **234 competency questions**, which cautions against a single query template (arxiv.org).

Published materialization research offers another quantitative precedent: one study examined **nine open ontologies** and introduced **five measures** for ontology-change effects (research.vu.nl). The lesson is methodological: report multiple deltas rather than compressing them into one pass percentage. Counts should include both denominators and exact comparison policies.

Release Evidence Package and Rollback

Make the packet self-contained

The release evidence package should let an independent reviewer reproduce the decision without resolving mutable “latest” resources. A minimum packet contains:

- **Candidate artifacts:** ontology, shapes graph, contexts, rules, and serializations.
- **Prior artifacts:** the exact release being replaced, not a regenerated approximation.
- **Import manifests:** requested IRIs, resolved IRIs, version IRIs, hashes, and local catalog mappings.
- **Test inputs:** positive, negative, boundary, historical, and migration fixtures.
- **Test definitions:** SHACL shapes, competency questions, SPARQL queries, consumer contracts, and tolerances.
- **Configurations:** processor and reasoner versions, profiles, flags, severity policy, and resource limits.
- **Outputs:** validation reports, consistency results, structural and entailment diffs, query results, and logs.
- **Change records:** human-readable notes plus machine-readable added, removed, replaced, split, and merged terms.
- **Migration evidence:** mappings, repair scripts, coverage counts, exception lists, and reversibility results.
- **Decision record:** status, scope, approver, time, unresolved unknowns, and evidence URIs.
- **Rollback target:** immutable previous bundle, resolver update, deployment instructions, and verification query.
- **Fixity data:** per-file checksums and a signed or otherwise controlled manifest where policy requires it.

PROV-O defines a bundle as a named set of provenance descriptions (w3.org). RO-Crate provenance guidance recommends retaining the old file version when recording a curation change (researchobject.org). The provenance record can describe evidence, but it does not choose the release policy.

Table 3 compares complementary evidence mechanisms, including the publisher's directly relevant system-of-record capability. None replaces an ontology reasoner, a SHACL processor, or consumer tests.

Mechanism	Primary role	What it contributes	What it does not establish
Versioned repository plus checksums	Artifact fixity	Exact files, immutable tags, diffable history, content digests	Logical consistency or consumer compatibility
PROV-O and DCAT metadata	Machine-readable lineage	Entities, activities, agents, revisions, distributions, version chains	Test correctness or approval policy
RO-Crate package	Portable research-object description	Self-describing attached package, tools, actions, and retained versions	Exhaustive manifest unless paired with fixity tooling (researchobject.org)

Mechanism	Primary role	What it contributes	What it does not establish
Reliable Context	Canonical evidence-bearing record	Verbatim, dated sources behind values, declared unknowns, and replayable past states	Ontology validation, reasoning, query regression, or migration execution
Release gate runner	Executable decision evidence	Processor outputs, diffs, query oracles, contract results, and status	Durable chain of custody unless outputs enter a governed evidence store

The mechanisms are layered, not interchangeable. RO-Crate recommends self-describing and self-contained attached packages, yet its metadata is not necessarily a full inventory (researchobject.org). It can interoperate with file-manifest and fixity mechanisms such as BagIt, Oxford Common File Layout, or Git (researchobject.org). The sound design assigns each layer a narrow claim.

Make rollback semantic, not merely infrastructural

Rollback has four coordinated targets:

- **Resolver rollback:** restore the current ontology IRI and distribution links to the prior immutable version.
- **Artifact rollback:** restore ontology, shapes, rules, contexts, imports, and catalogs as one closure.
- **Data rollback:** reverse or restore migrated data, including indexes and materializations.
- **Consumer rollback:** restore generated code, caches, query packs, and downstream configurations.

An infrastructure rollback alone is insufficient. Kubernetes, for example, keeps Deployment rollout history by default, but its documentation says only the Pod-template part is rolled back (kubernetes.io). A rollback rehearsal must therefore verify a semantic probe: resolve the expected version IRI, load the pinned closure, run a known ASK query, and confirm the expected result.

Version publication must also preserve the prior target. OBO Foundry requires consumers to be able to retrieve unaltered official release files in perpetuity (obofoundry.org). DataCite similarly recommends a new Digital Object Identifier for a major version while leaving the major-versus-minor judgment to stewards (support.datacite.org). The version identifier communicates the decision; the evidence packet defends it.

Implications and Future Directions

The immediate implication is organizational. Ontology release management is a **data-contract discipline**, not a final validation command. Ownership must cross ontology engineering, platform operations, data stewardship, and representative consumers. Each group owns different evidence: logical obligations, executable infrastructure, identifier and provenance policy, and observed behavior.

Three policy changes follow:

- **Define compatibility locally.** Name the signatures, queries, consumers, tolerances, and entailment regime that must remain stable.
- **Treat unknown as a decision state.** Missing imports, unavailable consumers, timeouts, and truncated reports remain visible and block unsupported compatibility claims.

- **Version evidence with artifacts.** Reports and manifests must be immutable, addressable, and retained for at least as long as the releases they justify.

Advance notification is part of migration evidence. OBO Foundry recommends announcing major changes to relevant stakeholders ahead of release and ordinarily gives a minimum of seven days (obofoundry.org). The exact window should be set by consumer lead times, not copied mechanically.

The standards landscape also requires status awareness. SHACL 1.2 is active draft work, while the stable Recommendation is an earlier specification. Draft capabilities such as version-aware shape imports and richer report context are useful, but release policy should pin the exact dated draft and processor behavior. Passing a standards test suite gives tested coverage, not proof of all untested behavior.

Emerging change vocabularies may improve machine-readable migration plans. The Knowledge Graph Change Language, for example, proposes a data model for ontology and knowledge-graph changes and represents both requested patches and observed diffs (arxiv.org). Such representations can connect release notes to executable migrations, but their presence still does not prove that a consumer contract passes. Earlier work also proposed invertible evolution mappings that aggregate low-level edits into complex changes such as merges and splits (arxiv.org). A separate provenance-aware approach addresses the human problem that raw diffs can contain hundreds or thousands of changes and can reconstruct the sequence leading to a resource state (cambridge.org).

Frequently Asked Questions (FAQs)

Does a conforming SHACL report mean the ontology is safe to publish?

No. It supports only the configured validation claim over the identified graphs. It does not establish OWL consistency, entailment stability, identifier continuity, import availability, query compatibility, or migration readiness. A processor failure is also distinct from a conforming result.

What is the minimum evidence for a compatible release?

At minimum: immutable candidate and prior artifacts, locked import closures, successful parse and lint, consistent reasoning result, expected SHACL fixtures, approved entailment and query diffs, passing consumer contracts, stable identifiers, and a rehearsed rollback. Every result should identify the tool and configuration that produced it.

When is an additive ontology change breaking?

It is breaking when it changes an obligation that a named consumer depends on. A new class can enter generated enumerations, a new domain axiom can add types, and a new shape target can expose existing data to constraints. “Additive” describes the artifact diff, not necessarily consumer behavior.

Should deleted terms disappear from the published ontology?

Usually not immediately. Deprecation preserves resolution and provides replacement guidance while consumers migrate. Published identifiers should not be reassigned. OBO's guidance requires logical axioms to be removed when a term is made obsolete, which also means the entailment delta must be tested (obofoundry.org).

How should saved SPARQL results be compared?

Compare normalized bindings as multisets unless the query defines order. Preserve duplicate multiplicities where they matter, compare RDF outputs by graph isomorphism or a declared canonicalization, and never compare blank-node labels directly. Record the active dataset and entailment regime.

What turns a failed gate into a breaking release rather than a stop?

An intentional, bounded delta plus complete migration evidence. The owner must identify affected consumers, publish mappings and repair steps, update expected results, demonstrate forward migration and rollback, retain the prior release, and obtain the accountable approval. Otherwise the outcome remains stopped.

Conclusion

Safe ontology release management begins by refusing to let one green indicator stand for the whole system. SHACL conformance is valuable, but it is a bounded observation over specific graphs and settings. The release decision must separately test nine gates: artifact integrity, SHACL fixtures, logical consistency, entailment change, identifier stability, import closure, queries and competency questions, consumer contracts, and migration and rollback.

The decisive distinction is between **artifact change** and **consumer-observable change**. Structural diffs locate edits. Reasoners locate logical effects. SHACL fixtures locate constraint effects. Queries and contracts locate operational effects. Version IRIs, immutable artifacts, manifests, and provenance make those observations reproducible. Migration and rollback evidence make an intentional incompatibility governable.

The resulting policy is simple enough to enforce. Publish as compatible only when every applicable gate passes under a declared compatibility contract. Publish as breaking only when the delta is intentional and a tested migration accompanies it. Stop whenever a material gate fails or remains unknown. That three-way decision is stricter than a boolean validation report, and it is also more honest: it states exactly what is known, what changed, and what downstream users must do next.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://www.w3.org/TR/shacl12-core
2. https://doi.org/10.1017/S0269888913000349
3. https://arxiv.org/abs/1804.07805
4. https://www.w3.org/TR/owl2-syntax

5. https://obofoundry.org/principles/fp-004-versioning.html
6. https://www.w3.org/TR/vocab-dcat-3
7. https://www.researchobject.org/ro-crate/specification/1.3/provenance.html
8. https://www.w3.org/TR/owl2-direct-semantics/original.html
9. https://research.vu.nl/en/publications/ontology-evolution-not-the-same-as-schema-evolution
10. https://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.8008.pdf
11. https://openric.org/spec/profiles/governance.html
12. https://robot.obolibrary.org/diff.html
13. https://obophenotype.github.io/uberon-robot-plugin/inference-diff.html
14. https://arxiv.org/abs/1401.5850
15. https://oboacademy.github.io/obook/lesson/developing-an-obo-ontology
16. https://obofoundry.org/principles/fp-019-term-stability.html
17. https://github.com/INCATools/ontology-development-kit/blob/master/docs/project-schema.md
18. https://www.w3.org/TR/sparql12-query
19. https://arxiv.org/abs/1811.09529
20. https://research.vu.nl/en/publications/beware-of-the-hierarchy-an-analysis-of-ontology-evolution-and-the
21. https://www.w3.org/TR/2013/PR-prov-o-20130312
22. https://www.researchobject.org/ro-crate/specification/1.3/structure.html
23. https://kubernetes.io/docs/concepts/workloads/controllers/deployment
24. https://support.datacite.org/docs/versioning
25. https://obofoundry.org/principles/fp-013-notification.html
26. https://arxiv.org/abs/2409.13906
27. https://arxiv.org/abs/1010.0122
28. https://www.cambridge.org/core/journals/knowledge-engineering-review/article/abs/evordf-evolving-the-exploration-of-ontology-evolution/348F681BDC1A6886795D0445A5BBF601

THE PUBLISHER

About Reliable Context

Reliable Context is a system of record for organizations that fuse private and public information and must defend every value they act on.

The guarantees

One canonical record per real thing: duplicates resolve by rule, ambiguity is shown rather than hidden, and related organizations relate without ever being merged. A verbatim, cited source and an observation date sit behind every value. Unknowns are declared as not found, blocked or not applicable with a reason, never left as an empty cell. Any past state of the record is reproducible, together with what changed since.

Where it runs

It runs inside the customer's perimeter, one database per organization. Permitted sources, model routing, reader clearance and retention are set as the customer's own policies, and every value is exportable with its chain of custody for an auditor or regulator.

What it is not

It is not a vector database or a RAG framework, not a data catalog, MDM or data-quality suite, not a CRM or data-enrichment vendor, and not a chatbot or AI assistant. Where a reader needs one of those, the honest answer is to say so and point them there.

Visit

See Reliable Context.

Reliable Context is in production use in two sectors; more than six thousand of its identity decisions were made by rule and none has been reversed. Articles published here explain general practice, cite their primary and academic sources, and name the jurisdiction of any regulatory rule they describe. They are not legal, regulatory or compliance advice for a specific organization, and requirements change.

Website: https://reliablecontext.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.