An ODRL Policy Bridge for SPARQL Access Control: Fail-Closed Compilation of Usage Policies into a Queryable Solid Access-Control View

Jesse Wright · the sparq project

Draft targeting the ESWC research track. Not submittable until the comparative study specified in §5.3 has been run and reported; artifact availability is stated in §8.

Abstract

Solid's access-control models answer *may this agent read graph G?*; ODRL usage control answers a richer question — *may this party use this asset, for purpose P, until time T, with obligation O discharged, disclosing only to recipient R?* Existing ODRL enforcement systems answer it with a separate policy engine evaluated per request. We compile instead: a matched ODRL rule is materialised into the very triples the host SPARQL engine's existing Web Access Control (WAC) / Access Control Policy (ACP) enforcement already understands — a Permission becomes an `auth:<mode>` grant in a queryable access-control view, a Prohibition an explicit `auth:deny<mode>` triple, and a faithfully-mappable recipient or time constraint a re-checked conditional grant rather than a frozen decision — so one decision surface, ordinary RDF answered by the engine's own SPARQL evaluator, serves WAC, ACP, and ODRL at once, and every compiled decision is queryable, provenance-tagged RDF (§4.6 walks one policy end-to-end, from Turtle to compiled triples to a SPARQL audit query and its result). The contribution is the fail-closed lifecycle discipline that compilation — unlike per-request evaluation — demands: materialise only definite decisions under a deliberately partial, narrowest-mode action map; realise ODRL's `prohibit` conflict strategy structurally through the target layer's unchanged allow-minus-deny set subtraction, disclosing that it is the only strategy implemented — a policy declaring a strategy the bridge cannot honour (`perm`, or `invalid` with a detected conflict) is loudly refused rather than silently coerced, while an unset `odrl:conflict` operates under `prohibit` rather than the ODRL-IM default `invalid` (§4.2, §6); retract asymmetrically on policy refresh, dropping stale grants eagerly while a materialised deny survives any re-evaluation that cannot prove the prohibition withdrawn; and consume stateful `odrl:count` budgets atomically. We position the design against the ODRL enforcement systems it is nearest to (OAC, ODRE, ODRL/Solid and UMA integrations) and, for the retraction rule, against the literature it actually lives in — authorization-decision caching, consistent-authorization staleness, and materialised-view maintenance. The evaluation is deliberately modest and we hold the line on it: the only direct evidence for the bridge today is four machine-checked answer-safety invariants over constructed scenarios; the CI-ratcheted WAC/ACP decision-parity floors of the target layer are context about the surface being compiled into, not bridge evidence; and the comparative decision-agreement study of §5.3 is specified but not run — this draft is not submittable until it has. No performance number appears, and no security, soundness, or cryptographic property is asserted.

1. Introduction

Two policy layers govern data in a Solid-style personal data store. The *access-control* layer (WAC's `.acl` documents, ACP's access-control resources — both Editor's Drafts of the W3C Solid Community Group, not W3C Recommendations) is enforced on every request by the host server or engine and answers mode-level questions: read, append, write. The *usage-control* layer (ODRL policies) is richer — parties, actions, targets, constraints, duties, prohibitions — but is conventionally evaluated by a separate engine, disconnected from the machinery that actually gates access. That disconnect has a cost: two evaluators to trust, two semantics to keep aligned, and usage-control decisions that are ephemeral (an evaluator's yes/no) rather than inspectable artefacts.

This paper describes a bridge built on a different principle: **compile, don't co-evaluate**. The host engine already maintains a queryable access-control *view* — a named graph of `principal auth:<mode> target` triples (and `auth:deny<mode>` duals) materialised from WAC/ACP sources, against which a single decision procedure computes each principal's accessible set as union-of-allow minus union-of-deny. The bridge evaluates an ODRL request fail-closed and, on a definite Permit or matched Prohibition, appends the corresponding grant or deny triple to that same view, mirrored into a provenance graph so bridged triples remain structurally distinguishable from static ones. No new enforcement engine is introduced; the existing view is the single decision surface, and because it is ordinary RDF in the engine's own store, *the decisions themselves are queryable* — §4.6 shows the artefact rather than asserting it.

Compilation is not free. A per-request evaluator faces each question fresh; a compiled view must answer three lifecycle questions that, to our knowledge, the nearest prior systems do not address head-on (§3): What happens when the underlying policy *changes* — in particular, when a prohibition's continued applicability becomes *ambiguous*? What happens to constraints whose truth varies per session (recipients, time windows) after the decision is frozen into a triple? And what happens to constraints that are genuinely *stateful*, such as an `odrl:count` budget? Our answers form a single fail-closed discipline, and they are the contribution.

Contributions. Each is refutable and forward-references its evidence; none is a performance claim, none claims a novel usage-control semantics, and none asserts a security, soundness, or attestation property. This is a research-track submission-in-preparation about a design point and its lifecycle discipline — it is *not* an in-use contribution: the artefact is a library-level, single-node integration with no deployment, no HTTP wire, and no users, and we do not claim otherwise.

- **C1 — Compilation into an existing queryable decision surface** (§4.1, worked end-to-end in §4.6). A matched ODRL Permission/Prohibition materialises as grant/deny triples in the host's existing WAC/ACP view, under a deliberately partial, narrowest-mode action mapping (`odrl:use` is refused, not widened). One decision procedure serves three policy languages, and decisions are auditable RDF with bridged-versus-static provenance separation.
- **C2 — The `prohibit` conflict strategy by construction** (§4.2). Given ODRL's `prohibit` (prohibition-overrides) strategy, the bridge realises it without a bespoke resolver, by the target layer's unchanged allow-minus-deny set subtraction; asserted as a machine-checked invariant (holds: `true`). It is the only strategy implemented: a policy declaring a strategy the bridge cannot honour (`perm`, `invalid` with a detected conflict, or an unrecognised value) is loudly refused, materialising nothing, and an *unset* `odrl:conflict` operates under `prohibit` rather than the ODRL-IM default `invalid` — a first-class limitation (§4.2, §6), not a footnote.
- **C3 — Asymmetric, fail-closed retraction on policy refresh** (§4.3). Stale grants are dropped eagerly; a materialised *deny* is retracted only when re-evaluation *proves* the prohibition withdrawn — an ambiguous re-evaluation keeps the deny, so refresh can never silently widen access (holds: `true`). We argue this polarity-asymmetric, three-valued retraction rule is the load-bearing novelty of the compiled approach, and we position it against the decision-caching and view-maintenance literature where a PC should look for it (§3).
- **C4 — Faithful dynamic and stateful constraints** (§4.4–4.5). Recipient and time constraints persist as per-session re-checked conditional grants; unmappable constraints fall back to a one-shot evaluation rather than being mis-encoded (holds: `true`); a stateful `odrl:count` budget is consumed atomically, isolated per (rule, party, target), with denials burning no budget (holds: `true`).
- **C5 — An honest evaluation frame** (§5). We state plainly what the evidence is: the direct evidence for C1–C4 today is the four machine-checked invariants of §5.1 — existence proofs over constructed scenarios, nothing stronger. The CI-ratcheted WAC (12-scenario) and ACP (12-scenario) decision-

parity floors of §5.2 evaluate the *pre-existing target layer*, not the bridge, and are cited as context only. The comparative decision-agreement study against independent ODRL enforcers is specified in §5.3 but has not run; this draft is not submittable until it has.

Non-claims. The bridge is a single-node library integration, not a new semantics and not a deployed system; the federated ODRL-to-MPC disclosure and ODRL-Duty-to-ZK obligation composition sometimes associated with this line of work is *designed only*, unbuilt, and out of scope (§7). No wall-clock number appears in this paper: the project's paper factory injects every figure from an evidence file whose headline channel refuses non-canonical (work-box) measurements by construction.

2. Related Work

ODRL and its normative landscape. The ODRL Information Model 2.2 and ODRL Vocabulary & Expression 2.2 are W3C Recommendations (2018). On conflicts the Information Model is precise, and we quote it precisely because an earlier draft of this paper did not: a Policy's `odrl:conflict` property takes one of `perm` (Permissions override Prohibitions), `prohibit` (Prohibitions override Permissions), or `invalid` (the entire Policy is void if a conflict is detected), and *"If the `conflict` property is not explicitly set, the default of `invalid` will be used."* Prohibition-overrides is therefore one selectable strategy — not the specification default. (The term "deny-overrides" is XACML's, where it names one of the standard rule- and policy-combining algorithms; ODRL's analogue is the `prohibit` strategy.) The ODRL Formal Semantics document of the W3C ODRL Community Group — a draft CG report, not a Recommendation — fixes rule activation, but its conflict-resolution machinery is explicitly marked pending at the time of writing, so no conflict default can be attributed to it. §4.2 states exactly which strategy the bridge implements and what that leaves out.

OAC — the ODRL profile for access control. Esteves, Rodríguez-Doncel, Pandit, Mondada and McBennett define OAC, an ODRL profile aligning ODRL with Solid access grants, with a policy editor and a matching algorithm: policies live in the pod and a matching step evaluates each access request against them, per request. OAC is the nearest *vocabulary-level* prior art, and the bridge is deliberately profile-compatible in spirit: like OAC it maps ODRL actions onto Solid access modes. It differs in *where the decision lives*: OAC-class systems answer from a separate evaluation over the policy graph; the bridge compiles the answer into the enforcement layer's own view, so no second evaluator sits on the request path and the materialised decision is itself queryable RDF. Because OAC materialises no decisions, the lifecycle problems this paper is about — retraction under ambiguity, per-session re-checking of frozen constraints, stateful budgets — do not arise in its published design, and it does not discuss them.

ODRE — enforcement-first ODRL. The Open Digital Rights Enforcement framework (Cimmino, Cano-Benito and García-Castro) argues ODRL policies should be *enforceable*, not merely descriptive: it integrates ODRL's descriptive terms with behaviour-specification languages (dynamic data handling, function evaluation) and ships open-source Python and Java engines that interpret enriched policies at request time. ODRE is the nearest *enforcement* prior art and the natural comparison system for §5.3. The architectural difference is the same as with OAC — ODRE brings its own enforcement engine, whereas the bridge re-uses the host's — and the lifecycle difference is that a per-request engine needs no retraction story at all: nothing is materialised, so nothing can go stale. That is a genuine simplicity advantage of ODRE's design, and we say so; the compiled approach buys a single decision surface and auditable decisions at the price of the refresh discipline of §4.3, which is exactly where our verification effort is concentrated. A persistent cross-request `odrl:count` budget of the kind §4.5 enforces is not described in the ODRE paper; its dynamic evaluation is exemplified on request-time (e.g. temporal) values.

ODRL/Solid and UMA integrations. Slabbinck et al. study the interpretation and evaluation of ODRL policies for Solid-style ecosystems, including a formally specified, interoperable ODRL evaluator with compliance reports and, earlier, a rule-based software agent enforcing usage-control policies over a Solid pod. That line established that precise ODRL evaluation alongside WAC/ACP is viable and is explicit about language mismatches; the bridge inherits its caution (unmappable constructs are refused or one-shot, never approximated) and adds the materialised-view lifecycle: refresh, asymmetric deny retraction, and stateful budgets. Slabbinck, Termont, Dedecker and Esteves take the opposite architectural route to ours: replace Solid's native access control with a User-Managed Access (UMA) authorization server that evaluates ODRL and issues tokens. The bridge is the dual trade — no new party and no token plane, decisions compiled down into the store's own access-control RDF; their design keeps decisions at a dedicated authorization service, ours makes them queryable data.

Materialised authorization decisions: caching, invalidation, and view maintenance. C3 is a retraction rule for a *materialised* authorization decision, so its real neighbours are not ODRL enforcers but three older literatures, and we position against them explicitly. *Authorization-decision caching* reuses PDP decisions at or near the enforcement point: SAAM (Crampton, Leung and Beznosov) infers approximate decisions from cached primary ones, and the authorization-recycling line (Wei, Crampton, Beznosov and Ripeanu) develops such caches with consistency techniques for hierarchical RBAC. Staleness there is a freshness/performance trade: the cache is an *optimisation* over an authoritative decision point that can always be re-consulted, so invalidation can be symmetric — evict and recompute. *Consistent authorization at scale* makes the staleness danger explicit: Zanzibar (Pang et al.) names the "new enemy" problem — serving access from ACL state that a causally earlier revocation removed — and prevents it with external-consistency tokens (zookies) over a versioned store. *Materialised-view maintenance* (Gupta and Mumick's survey; the delete-and-rederive algorithm of Gupta, Mumick and Subrahmanian) maintains a derived view under source updates by deleting affected derivations and re-deriving what still holds — which presumes re-derivation is decidable from the current sources. The bridge's setting breaks the shared assumption of all three in one specific way: the compiled view *is* the decision surface (no authoritative evaluator sits behind it on the request path to recompute from), and a refresh re-evaluation is not an oracle — it is three-valued, and an *ambiguous* outcome (a prohibition still structurally naming the request, with a constraint the refresh context cannot evidence) yields no recomputed truth to install. Symmetric evict-and-recompute is therefore unsound for denies: evicting a deny on ambiguity silently widens access — precisely a new-enemy admission, with no zookie protocol available because there is no consistent oracle to consult. Classical fail-safe defaults (Saltzer and Schroeder) do not decide this case either: they prescribe deny when *no* decision exists, but are silent on whether a *present* materialised deny whose justification has become unprovable may be dropped. C3's claim is exactly this gap: retraction of materialised authorization must be *polarity-asymmetric* under a three-valued re-evaluation — eager for grants, provably-withdrawn-only for denies. We have not found this rule stated in the caching, consistency, or view-maintenance literature, whose techniques (TTLs, causality tokens, delete-and-rederive) all assume a recomputable ground truth; §5.3's policy-mutation sequences are designed to probe exactly this rule empirically, and if the study surfaces a system that already embodies it, this claim will be withdrawn.

Feature-by-feature positioning. The table below summarises the differentiation. Qualitative cells describe the cited systems' published designs — verified against the cited papers, not measured; the §5.3 study operationalises the same comparison empirically, and its coverage matrix will replace our reading of the papers with per-construct measured rows.

Dimension	OAC-class (profile + matcher)	ODRE (enforcement engine)	This bridge
Decision surface	separate matching step over policy graph, per request	dedicated enforcement engine, per request	compiled into the host's existing queryable WAC/ACP view; no second engine on the request path
Conflict handling	profile-level; evaluator-resolved	engine-resolved at evaluation time	single `prohibit` strategy via the target layer's allow-minus-deny set subtraction (C2, machine-checked); unimplementable declared strategies loudly refused; unset default `prohibit`, not the ODRL-IM `invalid` (§4.2, §6)
Policy change / revocation	re-evaluate next request	re-evaluate next request (nothing materialised)	ledger + refresh; eager grant retraction; *asymmetric fail-closed* deny retraction under a three-valued re-evaluation (C3, machine-checked)
Session-varying constraints	evaluated at request time	evaluated at request time (incl. temporal)	persisted as per-session *re-checked conditional grants* (recipient, time); fail-open combinations refused; unmappable constraints fall back one-shot (C4)
Stateful constraints (`odrl:count`)	not addressed in the cited design (nothing persists across requests)	not described in the cited paper (dynamic evaluation is exemplified on request-time values)	atomic budget, isolated per (rule, party, target); denials burn nothing (C4, machine-checked)
Decisions auditable as RDF	policies are RDF; decisions ephemeral	policies are RDF; decisions ephemeral	decisions are triples in the auth view, mirrored to a provenance graph (worked example, §4.6)

Table 1: Qualitative positioning against the nearest prior art. "Machine-checked" cells are backed by the canonical invariants of §5.1; competitor cells are our reading of the cited papers' published designs (a threat noted in §5.4), to be replaced by the measured per-construct coverage matrix of the §5.3 study — no empirical superiority is claimed here or anywhere in this paper.

What we do *not* claim against any of these systems: broader ODRL construct coverage (ours is deliberately partial, §6), better performance (unmeasured), or a superior semantics (we implement one selectable conflict strategy of the standard model, §4.2). The claim is narrower and, we believe, defensible: compilation into an existing queryable enforcement surface is a distinct design point whose lifecycle obligations we identify and discharge fail-closed.

3. The Bridge

3.1. Compilation into the access-control view

The host engine governs graph access through an access-control *view*: a named graph (`<urn:sparq:auth>`) of `principal auth:<mode> target` triples and their `auth:deny<mode>` duals, materialised from WAC `.acl` documents or ACP access-control resources. One decision procedure computes a principal's accessible set as the union of allow-grants minus the union of deny-grants. Because the view is RDF in the engine's own store, it is queryable by the same SPARQL evaluator that answers user queries.

The bridge re-uses that surface. An ODRL evaluator answers a usage-control request — party, action, target, satisfied constraints, discharged duties — with a fail-closed allow/deny decision. On a definite Permit, the bridge maps the request's *concrete* ODRL action to the *narrowest* access mode it denotes: `odrl:read` / `display` / `present` / `print` / `play` to read; `odrl:append` to append; `odrl:modify` / `delete` / `write` to write. The `odrl:use` umbrella is deliberately left *unmapped*: materialising it as any single mode would have to pick the widest and violate fail-closedness, so a `use`-level request is refused rather than widened. A matched Prohibition is the dual: it appends the explicit `auth:deny<mode>` triple the decision procedure already subtracts. Each bridged triple is mirrored into a provenance graph, so a compiled decision stays structurally distinguishable from a static WAC/ACP one — this is what makes the audit query "which of this principal's rights came from ODRL?" a plain SPARQL query (shown concretely, with its result, in §4.6).

A grant is materialised *only* on a definite Permit whose action maps to a concrete mode and whose request names a concrete party (a WebID) and a target graph IRI. A Deny, an ambiguous evaluation, an unmapped action, or a missing party/target materialises *nothing*.

3.2. The `prohibit` conflict strategy by set subtraction

When a Permission and a matched Prohibition apply to the same principal, action, and target, the materialised deny is the operative decision: the unchanged decision procedure computes allow-set minus deny-set, so the deny removes the mode regardless of any allow grant.

We state the normative standing of this behaviour precisely, because an earlier draft of this paper mis-stated it. ODRL IM 2.2 defines the policy-level `odrl:conflict` property with three values — `perm` (Permissions override), `prohibit` (Prohibitions override), and `invalid` (the entire Policy is void if any conflict is detected) — and specifies `invalid` as the default when the property is not set. The ODRL Formal Semantics CG report does not supply a conflict default either; its conflict-resolution machinery is explicitly pending. Prohibition-overrides is therefore *a* strategy the specification admits, not "the ODRL default".

The bridge implements exactly one strategy — `prohibit` — structurally, because the target layer already implements set subtraction. It *does* consult `odrl:conflict`, as a fail-closed admissibility guard run before anything is materialised: a policy declaring a strategy the bridge cannot faithfully honour — `perm` (set subtraction is unidirectional, so a deny always wins), `invalid` with a detected permission/prohibition conflict (the bridge cannot void a whole policy), or an unrecognised `odrl:conflict` value — is loudly *refused*, materialising neither grants nor denies, rather than silently coerced into deny-overrides. (A declared `invalid` with no detected conflict has nothing to void and is admissible.)

What remains, and we disclose it rather than bury it, is the *unset* default: a policy declaring no `odrl:conflict` is processed under `prohibit`, not the specification default `invalid`. For a conflicting-yet-undeclared policy the bridge therefore materialises the uncontested rules and denies the contested (principal, mode, target) combinations, which is *more permissive* than voiding the whole policy as the Recommendation's default demands — though never fail-open: deny-overrides never grants a contested mode. This divergence is deliberate (honouring `invalid` for every undeclared conflicting policy would refuse the bridge's core deny-overrides use case), it is documented in the policy crate's conformance note, and it is carried as the *first* limitation in §6. The request-free static conflict lint (`detect_conflicts`) remains available for deployments that want to refuse conflicting policies up front.

What remains — and it is narrower than the earlier draft claimed — is C2: *given* the `prohibit` strategy, the bridge realises it with no bespoke resolver code to verify, and the property is machine-checked. With both a Permission and a matching Prohibition materialised for the same principal, that principal's accessible set for the contested mode is empty (holds: `true`).

3.3. Refresh and asymmetric fail-closed retraction

The harder direction — and, we argue, the part of the design that a per-request evaluator never has to get right — is *revocation*. Materialised triples are tracked in a ledger so that when the underlying policy changes, the bridge can refresh the view. On refresh, each tracked rule is re-evaluated. For *grants*, retraction is eager: an entry that no longer produces a grant (a withdrawn permission, a lapsed time window, a now-denying re-evaluation) is dropped, so withdrawn access is gone at the next decision.

For *denies*, eager symmetric retraction would be unsound: a re-evaluation can be *ambiguous* — unable to prove the prohibition either still applicable or definitely withdrawn — and retracting a deny on ambiguity silently restores access. The bridge therefore refines re-evaluation to three values (*applies* / *ambiguous* / *withdrawn*) and retracts a deny only on *withdrawn*. An ambiguous outcome keeps the deny. The asymmetry is deliberate: for grants, the fail-closed direction is to drop; for denies, it is to keep. Both halves compose — the subtraction property of §4.2 still holds after any refresh — and the retraction rule is asserted as a machine-checked invariant (holds: `true`).

3.4. Conditional grants: re-check what varies, refuse what would fail open

Many ODRL constraints vary per session and cannot be soundly frozen into a one-shot decision. The bridge partitions them.

Faithfully-mappable constraints persist as ACP *conditional grants* re-checked at decision time. A recipient constraint persists as a condition whose agent is re-checked per session, so only the named recipient is granted — the materialising party is not auto-granted; a recipient *set* becomes one condition per member, and a recipient *exclusion* becomes a public grant with an explicit carve-out. A `dateTime` window persists as live-clock bounds re-checked against the session clock, so a lapsed window denies immediately without waiting for a refresh.

Fail-open combinations are refused. For a *deny*, time windows are forbidden on the conditional path: a lapsed conditional deny would fail *open*, so the bridge forces a one-shot evaluation instead.

Unmappable constraints fall back safely. A constraint the bridge cannot map faithfully (for example `odrl:purpose`) is *not* encoded as a re-checked condition that would silently mis-judge it; it falls back to a one-shot evaluation, checked once and frozen — and §6 is explicit that this freezing is a real limitation, not a feature. The partition discipline is asserted as a machine-checked invariant — named recipient granted, materialising party not, unmappable constraint routed one-shot (holds: `true`).

3.5. Stateful constraints: atomic count budgets

An `odrl:count` budget needs genuine state, not a re-check of static data. Behind an opt-in count-enforcement feature, the bridge routes the grant through an evaluator that *atomically* consumes exactly one unit of budget per grant: it grants up to the limit and then denies, a denial consumes nothing, and budgets are isolated per (rule, party, target) — one party's exhaustion cannot starve another's, and probing denials cannot burn a budget. The property is asserted as a machine-checked invariant over the integer counter (holds: `true`). The counter store is pluggable (in-memory; an OS-lockfile-guarded file for cross-process use; a backend trait) — an engineering choice we do not present as a contribution.

3.6. A worked end-to-end example

The "decisions are queryable RDF" claim deserves one concrete artefact rather than repeated assertion. Take a policy under which alice may read a note but must not modify it:

```
@prefix odrl: <http://www.w3.org/ns/odrl/2/> .

<urn:policy/ex> a odrl:Set ;
  odrl:permission [
    odrl:action   odrl:read ;
    odrl:target   <https://pod.example/notes/n1> ;
    odrl:assignee <https://alice.example/card#me> ] ;
  odrl:prohibition [
    odrl:action   odrl:modify ;
    odrl:target   <https://pod.example/notes/n1> ;
    odrl:assignee <https://alice.example/card#me> ] .
```

Presented with alice's `odrl:read` and `odrl:modify` requests, the bridge evaluates each fail-closed and compiles the two definite outcomes into the view: the Permission becomes a grant triple on the read mode, and the Prohibition becomes an explicit deny triple on the write mode (`odrl:modify` maps to write under §4.1′s narrowest-mode table; mode granularity is the target layer's, so the deny covers the write mode as a whole). Each compiled triple lands in the enforcement graph and is mirrored verbatim into the provenance graph (TriG; `auth:` is the target layer's authorization vocabulary):

```
@prefix auth: <https://sparq.dev/ns/auth#> .

# the enforcement view — read by the unchanged decision procedure
<urn:sparq:auth> {
  <https://alice.example/card#me> auth:read      <https://pod.example/notes/n1> .
  <https://alice.example/card#me> auth:denyWrite <https://pod.example/notes/n1> .
}
# the provenance mirror — membership here marks a triple as bridged, not static
<urn:sparq:auth-bridged> {
  <https://alice.example/card#me> auth:read      <https://pod.example/notes/n1> .
  <https://alice.example/card#me> auth:denyWrite <https://pod.example/notes/n1> .
}
```

The audit question — *which of alice's rights were compiled from ODRL, rather than granted by a static WAC/ACP document?* — is now a plain SPARQL query over the store, answered by the same evaluator that serves user queries, with no bridge API involved:

```
PREFIX auth: <https://sparq.dev/ns/auth#>
SELECT ?right ?target WHERE {
  GRAPH <urn:sparq:auth>         { <https://alice.example/card#me> ?right ?target }
  GRAPH <urn:sparq:auth-bridged> { <https://alice.example/card#me> ?right ?target }
}
```

and returns exactly the two bridged decisions:

?right	?target
auth:read	<https://pod.example/notes/n1>
auth:denyWrite	<https://pod.example/notes/n1>

Table 2: Result of the audit query. A static WAC/ACP grant appears in <urn:sparq:auth> but not in the provenance mirror, so the join isolates bridged decisions.

One honest boundary, surfaced by writing this example down: the provenance mirror distinguishes *bridged from static*, not *which rule* produced a triple. Per-rule attribution exists at materialisation time (the evaluator reports the matched rules) and in the bridge's refresh ledger, but it is not currently materialised as RDF — so "which ODRL rule granted this right?" is answerable in-process but not yet by SPARQL alone. We record this as a limitation (§6) and tracked future work, not a delivered feature.

4. Evaluation

We separate what is evidence for the bridge from what is merely context. §5.1 is the only *direct* evidence C1–C4 have today: four machine-checked answer-safety invariants — existence proofs over hand-built scenarios, and we do not dress them up as more. §5.2 is *context*: the conformance floors of the pre-existing target layer, which evaluate the surface the bridge compiles into, not the bridge. §5.3 is the comparative study this draft specifies but has *not run* — the evaluation this contribution actually requires, and the reason the paper is not submittable yet. §5.4 states threats to validity. There is no performance tier: the project's methodology distinguishes canonical (deterministic, machine-independent) evidence from indicative work-box measurement, this paper's build refuses non-canonical numbers in headline positions by construction, and no canonical performance runner result exists for the bridge — so no timing appears anywhere in this paper.

4.1. Direct evidence: machine-checked answer-safety invariants

Four invariants pin the fail-closed discipline of §4. Each is a deterministic, CI-enforced assertion over the composed system (bridge + unchanged enforcement layer), injected here from the project's canonical evidence channel rather than transcribed by hand:

Invariant	Section	Holds
Prohibit-strategy subtraction through unchanged enforcement	§4.2	`true`
Asymmetric fail-closed deny retraction on refresh	§4.3	`true`
Recipient conditions re-checked per session; unmappable ⟶ one-shot	§4.4	`true`
Atomic, isolated, deny-neutral count budgets	§4.5	`true`

Table 3: Answer-safety invariants of the bridge — the whole of the direct evidence for C1–C4 in this draft. Honest framing: these are existence proofs over constructed scenarios — they demonstrate the discipline holds where it is exercised, and they are regression-guarded in CI, but they are not a completeness result and not an evaluation on third-party policies (that is §5.3, pending).

evidence: `crates/sparq-solid/tests/odrl_bridge.rs::deny_overrides_allow_through_enforcement` (and `permit_plus_prohibition_same_subject_is_denied`) (environment: canonical)

We are explicit about epistemic weight, because an earlier draft of this paper over-weighted this tier: an invariant of this kind refutes the claim "the bridge can be driven to widen access in scenario class X" for the constructed X, and nothing more. Their value is that each encodes exactly one clause of the fail-closed discipline, so a future regression in any clause fails CI loudly.

4.2. Context, not bridge evidence: conformance of the target layer

Nothing in this subsection evaluates the bridge, and none of it is evidence for C1–C4; we include it because the bridge's output is only as trustworthy as the decision layer it compiles into, and a reader

deserves to know what bounds *that* layer. The target layer carries deterministic, CI-enforced decision-parity ratchet floors — per-construct scenario corpora whose allow/deny decisions must match our reading of the WAC and ACP Editor's Drafts, with a floor that may only rise:

Suite	Ratchet floor	Basis
Solid WAC decision parity	12	per-construct `.acl` allow/deny scenarios (library-level)
Solid ACP decision parity	12	per-construct ACR allow/deny scenarios (library-level)

Table 4: Decision-parity floors of the pre-existing access-control layer the bridge materialises into — context for §4, not an evaluation of it. Two rows of the project's 7-suite cross-family conformance scoreboard (total floor 3442). Library-level decision parity only — not HTTP wire conformance, and not a security property.

evidence: `crates/sparq-solid/tests/conformance_wac.rs::WAC_SCENARIO_FLOOR` (mirrored in `crates/sparq-conformance/src/scoreboard.rs`; guarded by `tests/scoreboard_floors.rs`) (environment: canonical)

The same context framing applies upstream of the bridge: the ODRL evaluator whose decisions the bridge compiles is itself ratcheted against the third-party SolidLab ODRL Test Suite at a pinned, rise-only floor. That bounds the evaluator's agreement with an independent reading of ODRL — useful context — but it, too, says nothing about the compilation lifecycle that is this paper's contribution. Honest framing for both: these corpora are small, they test *readings* of drafts and vocabularies rather than wire behaviour, and they bound the layers the bridge composes with, no more.

4.3. The specified — not yet executed — comparative study

The gap this draft is honest about: no comparison against the direct competitors has been run, and without it this paper does not meet the research-track bar it targets. Rather than gesture at future work, we specify the protocol in full so it is falsifiable before it is run. One vocabulary correction against an earlier draft: we call this protocol *specified*, not *pre-registered* — pre-registration means a public, timestamped registry deposit made before data collection, and no such deposit exists; if one is made before the study runs, we will cite it here.

- **Systems.** The bridge (materialised-view path) versus the ODRE reference enforcement engines (Python and Java), and versus an OAC-style matcher where a runnable artefact is available.
- **Corpus.** Policies drawn from the cited systems' own published examples and test suites plus the SolidLab ODRL Test Suite cases, normalised into (party, action, target, context) request scenarios; the corpus and normalisation script do not exist yet and will be published with the results (§8).
- **Measure.** Per-request decision agreement: agree / disagree / out-of-scope-for-system, with every disagreement classified as semantics divergence, coverage gap (e.g. our unmapped `odrl:use`), or lifecycle difference (only the bridge has a refresh path to exercise; the study includes policy-mutation sequences to probe exactly the §4.3 asymmetry, including its claim to novelty — §3).
- **Coverage matrix.** Per ODRL construct (actions, constraint left-operands, duty forms, conflict strategies): supported / partial / refused, for each system, replacing §3′s qualitative competitor cells with measured rows — including the bridge's own refused rows (`odrl:use`; `odrl:conflict perm`/`invalid`).
- **Explicit non-goals.** No latency or throughput comparison unless and until a canonical (bare-metal, pinned) runner exists; work-box timings would be non-canonical and are excluded by the paper factory's honesty gate regardless.

Executing this protocol is the single blocking item between this draft and submission (tracked in the project's task system as the sq-gum8.3 rewrite program); the registry status of this paper remains `draft`, and we hold that line rather than submit an evaluation-free systems paper.

4.4. Threats to validity

Construct validity. The §5.1 invariants are existence proofs over constructed scenarios; passing them does not certify the absence of an unsafe grant outside the constructed classes. *Internal validity.* The §5.2 context floors are judged against our own encoding of the WAC/ACP Editor's Drafts — a shared misreading would pass both the corpus and the bridge. *External validity.* Everything verified is verified on one engine and one implementation of the bridge; nothing yet says the compiled-view design point transfers, and until §5.3 runs there is no evidence about behaviour on third-party policies. *Selection bias.* The §3 comparison table was authored by us from the cited systems' papers; the pending study is designed to replace precisely those cells with measured values.

5. Limitations

Beyond the pending comparative study (§5.3), seven limitations bound the claim.

First — and normatively the most important — the bridge implements exactly one conflict strategy. ODRL's `prohibit` strategy is realised structurally, and a policy *declaring* a strategy the bridge cannot honour (`perm`, `invalid` with a detected conflict, or an unrecognised value) is loudly refused before anything is materialised (§4.2). What diverges is the *unset* default: a policy declaring no `odrl:conflict` is processed under `prohibit`, whereas the Recommendation's stated default is `invalid` — so a conflicting-yet-undeclared policy has its uncontested rules honoured rather than the whole policy voided, more permissive than the specification default (though never fail-open: the contested combinations are denied). The divergence is deliberate — deny-overrides is the fail-closed side of an authorization decision — and is documented as the conformance note in the policy crate.

Second, the action-to-mode mapping is deliberately conservative and partial: the `odrl:use` umbrella is unmapped, so a request must name a concrete action to be bridged. This is a fail-closed choice, not full ODRL action coverage, and it will show up as "refused" rows in the §5.3 coverage matrix.

Third, the constraints handled as re-checked conditions are exactly the faithfully-mappable recipient and time constraints; everything else — notably `odrl:purpose` — is one-shot, so a purpose decision is frozen at materialisation rather than re-checked. A purpose-heavy policy regime loses the main benefit of the conditional path.

Fourth, the refresh discipline (§4.3) is only as good as the triggers that invoke it: a policy change the host never surfaces to the bridge leaves stale grants until the next refresh. The eager-grant/asymmetric-deny rule bounds the damage direction (staleness can persist access it should have dropped only until refresh; it can never un-deny), but the window is real.

Fifth, provenance is bridged-versus-static only: the provenance mirror marks *that* a triple was compiled from ODRL, not *which rule* compiled it (§4.6). Per-rule attribution lives in the in-process decision report and the refresh ledger, not in the RDF, so rule-level audit is not yet a SPARQL query. Materialising rule-level provenance is tracked future work.

Sixth, this is a library-level, single-node surface: no HTTP wire conformance, no multi-server deployment, no users, and no claim that the view scales to adversarial policy volumes — unmeasured, per the no-non-canonical-numbers rule. This is also why the paper makes no "in-use" claim.

Seventh, nothing in this paper is a security result. The invariants are answer-safety properties of the composition under the stated model, not a penetration analysis, a proof of completeness, or a proof of the absence of an unsafe grant.

6. Deferred: the federated and cryptographic composition

An earlier framing of this work leaned on a federated composition — per-node ODRL policies drawing the disclosed-versus-hidden boundary for a multi-party computation, and an ODRL `Duty` compiling

to a zero-knowledge proof obligation. We state plainly that this half is *designed only*: it is unbuilt, it inherits a multi-party-computation envelope that is honest-majority and semi-honest over a low-latency network, and it would build on a zero-knowledge estate that is research-grade and has *not* been externally audited — the project's external-audit gate (bead sq-qhy4) is open, and the collaborative multi-prover path is itself under an open re-audit. We therefore claim *no* security, privacy, integrity, or attestation property for any federated or cryptographic disclosure, and this paper's contribution stands or falls on the single-node bridge alone. The deferred composition is mentioned only so that no reader infers it is implied by what is built.

7. Artifact and resource availability

The implementation, its tests, and this paper's source are open under the MIT license in the sparq repository (`github.com/jeswr/sparq`). The bridge is the opt-in `odrl-bridge` feature of the `sparq-solid` crate; the stateful count path is the opt-in `count-enforcement` feature of the `sparq-policy` crate; the four §5.1 invariants are ordinary `cargo test` regression tests run in that repository's required CI (the non-anonymous footer of this paper names the exact test files). Every figure in this paper is injected at build time from a committed evidence file (`site/src/data/paper-evidence.json`) whose records must be deterministic, machine-independent, and traceable to a named test; the Typst source of this paper is committed alongside it, and the PDF and the in-site HTML render from that single source. The third-party conformance suite used by the ODRL evaluator (the SolidLab ODRL Test Suite, MIT) is fetched by script, not vendored. Stated so the availability claim is no wider than the estate: the §5.3 comparative-study corpus and its normalisation script are *not yet staged* — they will be published alongside the study's results, before any submission. No DOI-archived snapshot has been minted for this draft; one will be minted at submission time, with an anonymised artifact variant for double-blind review.

8. Conclusion

A usage-control layer need not bring its own enforcement engine. A matched ODRL rule can be compiled into the very triples an existing, queryable WAC/ACP view already understands, so one SPARQL-backed decision procedure serves both layers and every usage-control decision is an auditable, provenance-tagged artefact (§4.6). The price of compilation is a lifecycle discipline, and that discipline is the contribution: fail-closed materialisation with a narrowest-mode partial action map, the `prohibit` conflict strategy by set subtraction with its non-representable `perm`/`invalid` strategies disclosed, asymmetric three-valued deny retraction on refresh — positioned against decision caching, consistent authorization, and view maintenance, not only against ODRL enforcers — re-checked conditional grants with safe one-shot fallback, and atomic count budgets, each machine-checked (§5.1). The evaluation is honest about its weight: four invariants are the only direct evidence, the target layer's conformance floors are context, and the specified comparative decision-agreement study of §5.3 is the acknowledged, blocking gap between this draft and submission. No wall-clock number is claimed, and no cryptographic property is asserted; the federated composition remains deferred behind an open external audit.

9. References

1. W3C. *ODRL Information Model 2.2*. W3C Recommendation, 15 February 2018. https://www.w3.org/TR/odrl-model/ (the `conflict` property: values `perm`, `prohibit`, `invalid`; default `invalid` when not set).
2. W3C. *ODRL Vocabulary & Expression 2.2*. W3C Recommendation, 15 February 2018. https://www.w3.org/TR/odrl-vocab/.

3. W3C ODRL Community Group. *ODRL Formal Semantics*. Draft Community Group Report (not a W3C Recommendation; conflict-resolution machinery marked pending as of this draft). https://w3c.github.io/odrl/formal-semantics/.
4. Esteves, B., Rodríguez-Doncel, V., Pandit, H. J., Mondada, N., McBennett, P. *Using the ODRL Profile for Access Control for Solid Pod Resource Governance*. In: The Semantic Web: ESWC 2022 Satellite Events, LNCS 13384, Springer, 2022. doi:10.1007/978-3-031-11609-4_3.
5. Cimmino, A., Cano-Benito, J., García-Castro, R. *Open Digital Rights Enforcement Framework (ODRE): from descriptive to enforceable policies*. arXiv:2409.17602, 2024 (preprint).
6. Slabbinck, W., Rojas Meléndez, J., Esteves, B., Colpaert, P., Verborgh, R. *Interoperable Interpretation and Evaluation of ODRL Policies*. In: The Semantic Web — ESWC 2025, LNCS, Springer, 2025. doi:10.1007/978-3-031-94578-6_11.
7. Slabbinck, W., Termont, W., Dedecker, R., Esteves, B. *From Access Control to Usage Control with User-Managed Access*. arXiv:2601.18761, 2026 (preprint).
8. Solid Community Group. *Web Access Control (WAC)*. Editor's Draft (not a W3C Standard, not on the W3C Standards Track). https://solid.github.io/web-access-control-spec/.
9. Solid Community Group. *Access Control Policy (ACP)*. Editor's Draft, 2022-09-29 (not a W3C Standard, not on the W3C Standards Track). https://solid.github.io/authorization-panel/acp-specification/.
10. OASIS. *eXtensible Access Control Markup Language (XACML) Version 3.0*. OASIS Standard, January 2013 (the deny-overrides / permit-overrides combining algorithms).
11. Saltzer, J. H., Schroeder, M. D. *The Protection of Information in Computer Systems*. Proceedings of the IEEE 63(9), 1975 (fail-safe defaults).
12. Crampton, J., Leung, W., Beznosov, K. *The Secondary and Approximate Authorization Model and its Application to Bell-LaPadula Policies*. In: SACMAT 2006, ACM. doi:10.1145/1133058.1133075.
13. Wei, Q., Crampton, J., Beznosov, K., Ripeanu, M. *Authorization Recycling in Hierarchical RBAC Systems*. ACM Transactions on Information and System Security 14(1), 2011. doi:10.1145/1952982.1952985.
14. Pang, R., Cáceres, R., Burrows, M., et al. *Zanzibar: Google's Consistent, Global Authorization System*. In: USENIX Annual Technical Conference (ATC), 2019.
15. Gupta, A., Mumick, I. S. *Maintenance of Materialized Views: Problems, Techniques, and Applications*. IEEE Data Engineering Bulletin 18(2), 1995.
16. Gupta, A., Mumick, I. S., Subrahmanian, V. S. *Maintaining Views Incrementally*. In: ACM SIGMOD 1993.
17. W3C. *SPARQL 1.1 Query Language*. W3C Recommendation, 21 March 2013.
18. SolidLab. *ODRL Test Suite* (MIT). https://github.com/SolidLabResearch/ODRL-Test-Suite.

---

sparq project · evidence traces to `crates/sparq-solid/src/odrl_bridge.rs` and its tests in `crates/sparq-solid/tests/odrl_bridge.rs`, the count-enforcement tests in `crates/sparq-policy/tests/odrl_count.rs`, and the Solid WAC/ACP decision-parity ratchets in `crates/sparq-solid/tests/conformance_wac.rs` / `conformance_acp.rs` (mirrored in the cross-family scoreboard `crates/sparq-conformance/src/scoreboard.rs`, drift-guarded by `tests/scoreboard_floors.rs`). Numbers in this document are injected at build time from the paper-bound evidence file; see the provenance stamp on the published page.