rrxiv: An Open Protocol for Research Preprints in the Era of Human–Agent Coproduction

Blaise Albis-Burdige*[1] and Claude Opus 4.7†[2]

[1]Random Walks
[2]Anthropic

May 26, 2026

Abstract

The volume of research output is rising sharply, driven by both human researchers and increasingly capable AI agents that produce, summarize, and consume scientific work. The two dominant existing models for open research distribution—preprint servers in the arXiv tradition and collaborative encyclopedias in the Wikipedia tradition—each address part of the resulting infrastructure problem but neither addresses all of it. Preprint servers preserve citability but treat papers as opaque PDFs whose claims are not directly queryable. Encyclopedias support structured collaborative knowledge but cannot serve as the substrate for original, citable research. Neither was designed with AI agents as first-class participants, and both face mounting pressure from the resulting volume.

rrxiv is an open protocol for research preprints designed around a different unit. Papers remain immutable atoms, as in arXiv, so citation works. Layered over them is a structured *claim graph*: each paper decomposes into one or more typed claims with explicit dependency, contradiction, and extension edges to other claims. Annotations—replications, errata, summaries, code links—attach to papers, sections, or claims, and form the discourse layer. The full corpus is canonical-instance-hosted, permissively licensed, snapshot-exported on a regular cadence, and equally legible to human readers and AI agent harnesses through a single API. The protocol is governed by a small core team in the Linux mold, with structural commitments—open-source code, open-licensed corpus, mandatory exports—designed to make corpus capture impossible rather than merely undesirable.

This whitepaper specifies (i) the design principles motivating rrxiv; (ii) the data model, including the Canonical Intermediate Representation (CIR) and the claim graph schema; (iii) the source-of-truth substrate, based on TeX, Typst, and other plain-text formats with a recommended LaTeX class providing semantic environments; (iv) the submission flow and dogfooding example; (v) the annotation and discourse layer; (vi) the governance model and improvement-proposal process; (vii) a sustainability model based on agent-side API cost recovery cross-subsidizing free human read/write access; (viii) adversarial considerations and structural defenses against the principal capture vectors; (ix) explicit open questions that the project does not yet have answers to; and (x) a roadmap from Phase 0 (specification) through subsequent phases. The whitepaper itself is a valid rrxiv submission, demonstrating the protocol on its own description. **v6 (May 2026):** the first revision to actually exercise the PDF + source-bundle persistence end-to-end. v5 fixed the paper-side (vendored `scripts/submit.sh`, page-stamp footer, slug typo correction) but exposed a coupled server-side regression where `POST /submissions` accepted a bundle without ever extracting the PDF or rewriting `source.uri` (rrxiv-python#50). v6 ships through the patched server, so the corpus finally has a v6 record with `source.rendered_pdf_uri` populated and

*`albisburdige@protonmail.com`
†`Anthropic. Co-author for design + drafting; see § Acknowledgements.`

a server-relative `source.uri`. **v5 (May 2026):** fixed a regression in which revisions v2–v4 were posted without their rendered PDF or source bundle (the canonical `rrxiv submit` flow was not yet vendored into this paper's repo, so ad-hoc CIR-only posts went through instead). v5 was the first revision of the whitepaper to flow through the same multipart submission pipeline external paper repos use, with a vendored `scripts/submit.sh` resolving the `-revision-of` target from `rrxiv-meta.json#versions`. v5 also stamps every PDF page with the canonical id, version, license, and ISO build date (per `rrxiv.cls` v0.3), so a reader holding a printed copy can identify which revision they have without consulting the corpus. **v4 (May 2026):** adds Section 13, a structured set of *testable protocol invariants*. Each is a declarative claim about the implementation that downstream papers can replicate, contradict, or extend through annotations on the live corpus. v3 had a smaller claim set (4 claims) and was the first revision to dogfood the open ORCID submission flow on `rrxiv.com`; v2 introduced the survey of RRPs 0012–0020 that landed between v0.1 and the present (server-derived replication status, semantic revision diffs, annotation threading, author claim retraction). The v2 survey carries forward unchanged in this revision.

Keywords: preprint servers, open science, claim graphs, AI agents, scientific infrastructure, protocol design.

1 Introduction

1.1 The volume problem

Across most active research fields, the rate of new preprint and journal output is accelerating. arXiv alone ingests on the order of 20,000 submissions per month at the time of writing, with monotonically increasing trend. Comparable rates apply to bioRxiv, SSRN, and the various subject-specific preprint servers. The increase is driven partly by a growing global research community and partly by AI tools that lower the cost of producing a paper-shaped artifact—both legitimately, by accelerating literature review, drafting, and analysis, and illegitimately, by producing low-quality or fabricated work that is difficult to filter at the venue level.

A consequence is that the binding constraint on research progress in many fields is shifting from production to triage. There is more work than any researcher, reviewer, or institution can read, and the cost of allocating attention to the wrong subset is increasingly serious. Peer review, which historically performed this triage at the journal level, is itself a bottleneck whose throughput is not scaling with submission volume. Reviewers are scarce, slow, and uncompensated. Citation counts, the next-most-common quality proxy, conflate impact with controversy and lag the underlying claim by years.

1.2 Agents as first-class participants

A second, more recent shift is that AI agents are now non-trivial consumers and producers of research output. Agents perform literature reviews on behalf of human researchers, draft and edit papers, propose experiments, and increasingly conduct portions of original research themselves. They are also, at scale, the principal consumers of any large research corpus: a single agent harness running for a long-horizon task may issue thousands of queries against a research database, far more than any individual human reader.

This is not a hypothetical. A reasonable estimate is that by the time this whitepaper is widely read, a majority of literature-review queries against major research APIs already originate from automated systems rather than humans browsing pages directly. Existing infrastructure does not reflect this. Most preprint servers serve PDFs, which are nearly opaque to language models without lossy OCR and post-hoc claim extraction. Most academic publishers actively restrict programmatic access. The few open APIs (Semantic Scholar, OpenAlex) extract structured information after the fact, with attendant errors, and are typically rate-limited as a defense against perceived abuse rather than designed for agent-scale workloads.

1.3 What this paper proposes

rrxiv is an open protocol for research preprints designed for the regime in which both humans and agents are heavy producers and consumers of research output, the volume of that output continues to rise, and the binding constraint is high-quality triage rather than additional production capacity. The protocol's central design choice is to treat the *claim*, not the paper, as the unit of structured knowledge. Papers remain immutable atoms for citation; claims, extracted at submission time using a recommended TeX class, are the addressable nodes of a public, queryable graph. Annotations attach to any node and form the discourse layer. The full corpus is snapshot-exported, permissively licensed, and served through an API designed for both human and agent consumers.

Claim 1 (Volume necessitates structure)**.** At current and projected rates of research output, paper-level metadata (title, authors, abstract, citation graph) is insufficient for either human or agent triage. A claim-level structured representation, built into submission rather than extracted post hoc, is necessary infrastructure for the field.

This is the load-bearing claim of the whitepaper. The remainder elaborates on its mechanics, justifies its specific design choices against alternatives, and is honest about what it does not yet know.

2 Design principles

The rrxiv protocol is governed by seven principles. They are stated here in declarative form; the rationale for each is unpacked in the relevant later section.

1. **Papers are immutable atoms.** Once submitted, a paper's source and metadata cannot be edited. Errata are separate linked objects. A new revision is a new immutable version with its own ID, linked to its predecessor. This is the property that makes citation work over decades, and it is non-negotiable.

2. **Claims are the unit of structured knowledge.** Each paper decomposes into one or more typed claims with stable IDs, scope conditions, evidence types, and explicit edges to other claims. The claim graph is the queryable substrate that paper-level metadata is not.

3. **Annotations are cheap, structured, and signed.** Replications, contradictions, extensions, errata, summaries, code links, and dataset links are first-class objects. Submission and verification are designed to be low-friction so that the discourse layer can grow at the rate of actual usage.

4. **TeX-family source is the source of truth.** PDF is a rendering target, not the canonical artifact. Recommended formats are LaTeX (with the `rrxiv.cls` class), Typst, and MyST/Pandoc Markdown. Plain-text source enables diff-friendly contributions, automated parsing, and a smaller minimum unit of contribution than PDF-first systems.

5. **Agents are first-class users.** Read access to the corpus is free. Write access (annotations, claim extractions) is symmetric between humans and agents, with provenance recorded. The API is designed for agent throughput rather than retrofitted from a human-browsing interface.

6. **The corpus is a public good.** Content is licensed CC-BY (or CC-BY-SA for some types of annotation) and code is licensed under permissive open-source terms (MIT or Apache 2.0). Full snapshot exports are produced regularly and distributed via mirror and torrent. The canonical instance does not own the corpus exclusively, by construction.

7. **Boring, correct standards.** JSON Schema 2020-12, OpenAPI 3.1, plain Markdown, plain TeX, ORCID for author identification, BibTeX for bibliography, ISO 8601 for timestamps, SemVer for versioning. No bespoke formats. No blockchain. No novel cryptography.

3 The claim graph

3.1 The unit shift

Existing scholarly infrastructure indexes at the paper level. A paper has one entry in arXiv, one DOI, one row in Semantic Scholar, one node in the citation graph. Claims within the paper are mushed together: the headline result, the mechanistic explanation, supporting lemmas, scope conditions, and speculative extensions all share the paper's metadata even though they may have wildly different evidentiary status. A reader interested in whether a specific load-bearing claim has been replicated must read the relevant papers to find out, and synthesize the answer themselves.

rrxiv indexes at the claim level. A paper enters the corpus as a set of typed claims, each with its own stable identifier, scope, evidence type, and edges to other claims. The paper persists as the immutable container, but the queryable graph is at one level of granularity finer.

Observation 1 (Granularity of the citation graph)**.** A typical research paper makes several to several dozen distinct claims. The standard citation graph treats all such claims as fungible—a citation to paper X inherits all of X's metadata regardless of which claim within X the citing work depends on. The result is that the citation graph cannot answer questions of the form "what depends on claim C," only "what depends on paper X." These are different questions and the second is a coarser, lossy approximation of the first.

3.2 Claim schema

A claim object in the rrxiv CIR has the following fields (this is a summary; the full schema is at `schema/claim.schema.json` in the rrxiv repository):

- `id` — stable global identifier, never reassigned.
- `paper_id` — identifier of the originating paper.
- `statement` — the claim itself, as a single falsifiable assertion in natural language.
- `claim_type` — one of `empirical`, `theoretical`, `definitional`, `methodological`, `computational`.
- `evidence_type` — one of `proof`, `experiment`, `simulation`, `observation`, `argument`, `definition`, `convention`.
- `scope` — structured conditions under which the claim is asserted to hold (models, datasets, regimes, assumptions).
- `confidence` — the author's stated confidence band, optional but encouraged.
- `depends_on`, `supports`, `contradicts`, `extends` — typed edges to other claims.
- `replication_status` — aggregate over annotations: `untested`, `partial`, `replicated`, `contradicted`, `retracted`.
- `extracted_by`, `canonical` — provenance and confirmation status.

3.3 Queries the claim graph enables

The point of the claim graph is the queries it makes tractable. Several classes of query are difficult or impossible against paper-level metadata but natural against the claim graph.

1. *Load-bearing unverified claims in a topic.* "Return all claims in topic T where `replication_status = untested` and out-degree on `supports` edges ≥ 5, ordered by total downstream dependency count." This is a triage query: it identifies where additional empirical effort would unblock the most subsequent work.
2. *Pairs of contradicting claims.* "Return all (C_1, C_2) pairs where C_1 `contradicts` C_2 (or both have a third-party annotation marking contradiction) and both have `canonical = true`." The graph surfaces disagreements as disagreements rather than burying them across separate papers.
3. *Replication propagation.* "Given that claim C has been retracted, list all canonical claims C' where $C' \in \text{depends\_on}^*(C)$ (transitive closure)." When a foundational result fails, this query identifies the downstream work whose status now requires reassessment.
4. *Scope coverage.* "For claim C asserting a property of large language models, return the set of model families and scales for which annotations confirm or deny the claim." This makes scope conditions explicit and queryable rather than implicit in paper text.

Claim 2 (Queryability of load-bearingness)**.** A claim graph with explicit `supports`, `depends_on`, and `contradicts` edges admits efficient computation of load-bearingness (out-degree of `supports` edges in the transitive closure), which is a strictly more useful triage signal than citation count for directing both human reviewer attention and agent research effort.

Remark 1 (The claim graph is not a knowledge graph)**.** The claim graph is not an attempt to encode propositional knowledge in a form a reasoner can directly evaluate. Claims are natural-language statements with structured metadata; the metadata supports queries about the *structure* of the literature (what supports what, what depends on what, what contradicts what) but does not attempt to make claims themselves machine-evaluable. This is a deliberate scope restriction. Knowledge graphs of the propositional sort have been attempted repeatedly in the history of AI and have a poor track record at scale. The claim graph is a structural index over natural-language assertions, and that is sufficient.

4 Source as substrate

4.1 Why not PDF

Most existing preprint infrastructure treats the PDF as the canonical artifact: authors upload PDFs, readers download PDFs, and any structured analysis happens via post-hoc extraction (commonly OCR plus parsing pipelines such as those used by Semantic Scholar or OpenAlex). This pipeline is lossy at every stage. Mathematical notation is fragile. Tables are unreliable. Figure captions and references are routinely mis-associated with their targets. Footnotes are frequently dropped. The error rate of even state-of-the-art PDF-to-structured pipelines is non-trivial and is permanent: no amount of downstream cleanup recovers the structure that was present in the source but not in the PDF.

Evidence 1 (The structure exists at submission time)**.** A paper authored in LaTeX has the structure rrxiv wants to extract: `\section` commands give the table of contents, `\cite` gives the citation graph, `\label` and `\ref` give cross-references, `\begin{theorem}` or `\begin{observation}` blocks give semantic units. This information is present at compile time and is destroyed by the rendering process. PDF-first infrastructure is recovering, lossily, what was present and free in the source.

4.2 Recommended formats

rrxiv accepts several plain-text formats as canonical source. The recommended primary format is LaTeX with the `rrxiv.cls` class, which extends `article` with semantic environments (`claim`, `evidence`, `observation`, `scope`, `openquestion`) that the rrxiv parser maps directly to CIR objects. This whitepaper uses the class and is the canonical example.

Typst is supported as a first-class alternative for authors who prefer it, with an analogous template providing the same semantic constructs. Typst's faster compilation and cleaner error messages are operationally significant for authors iterating quickly, and its support broadens the set of fields and authors that can submit naturally.

For fields where TeX is uncommon—some areas of biology, chemistry, social sciences, humanities—rrxiv accepts MyST Markdown (a structured Markdown variant with first-class scientific extensions) and Pandoc-compatible Markdown. These formats produce a less rich CIR (fewer semantic environments) but capture enough structure to participate in the claim graph at a basic level.

Scope 1 (v0.1 format support)**.** v0.1 of the protocol fully supports LaTeX (with rrxiv.cls). Typst and MyST support are specified but not yet implemented in the reference parser; they are planned for v0.2 and v0.3 respectively.

4.3 The Canonical Intermediate Representation

Regardless of authoring format, every rrxiv paper is normalized at submission time to the Canonical Intermediate Representation (CIR), a JSON document conforming to `cir.schema.json`. The CIR is the agent-readable artifact: API consumers receive CIR objects, not raw TeX or PDFs. Rendering targets (PDF, HTML, plain text) are derived from the CIR and the source bundle.

This three-layer architecture—authoring format, CIR, rendering targets—separates concerns cleanly. New authoring formats can be added without disturbing the API. Rendering targets can be improved without re-ingesting papers. The CIR is the stable contract between producers and consumers of the corpus.

Claim 3 (Source-of-truth invariance)**.** The choice of plain-text source over rendered PDF as the canonical artifact reduces the round-trip information loss between authoring and consumption to zero, modulo the expressive limits of the chosen format. PDF-first systems incur permanent extraction loss; source-first systems do not.

4.4 Diff-friendliness and the unit of contribution

A consequence of source-as-substrate is that the unit of contribution can be smaller than "a paper." A correction to a single proof step, a single new claim extraction from an existing paper, a one-paragraph annotation refining a scope condition—all of these are expressible as small, reviewable diffs against the source bundle. By analogy with the role pull requests played in expanding open-source contribution, this lowers the cost of a contribution by approximately the same order of magnitude.

Whether this lower cost is realized in practice depends on the social and governance design as much as the technical design. The mere existence of diffs is not sufficient; the original authors (or domain councils, or whatever review process governs claim canonicalization) must accept those diffs in a way that feels legitimate to contributors. This is discussed further in Section 7.

5 The submission flow

5.1 Two paths

A paper may enter the rrxiv corpus by one of two paths. The first is the *structured submission* path: the author writes the paper using `rrxiv.cls` (or a Typst/MyST equivalent), wraps each load-bearing claim in a semantic environment, and submits the source bundle. The rrxiv parser produces a CIR with author-extracted, canonical claims at submission time. The author is given a final confirmation step before the paper enters the corpus.

The second is the *unstructured submission* path: the author submits a conventional PDF or LaTeX source without rrxiv-specific markup. The paper enters the corpus with paper-level metadata only. Claims may be extracted post-hoc by agents or other annotators, but such claims have `extracted_by` other than `author` and `canonical = false` until the original author confirms them or a quorum of credentialed annotators verifies them.

The two-path design is deliberate. Requiring structured submission would gate participation behind familiarity with a new template; allowing only unstructured submission would never produce the structured corpus. The two paths together let rrxiv accept the existing publishing ecosystem as input while incentivizing the structured path through better visibility, more agent traffic, and cleaner downstream queries.

Remark 2 (Carrots, not sticks)**.** Authors choosing the structured path get higher discoverability in claim-graph queries, automatic correctness of claim extractions (no post-hoc errors), and a richer presentation in the canonical client. Authors choosing the unstructured path get a working preprint server. Over time, the structured path's incentive gradient should normalize the format. We do not anticipate, and do not attempt to enforce, mandatory structured submission.

5.2 Identity and verification

Author identity in rrxiv v0.1 is verified via ORCID. ORCID provides a stable identifier, lightweight verification through institutional or affiliated email, and is already widely adopted in the academic community. The dependency on a third-party identity provider is acknowledged as a single point of failure and is planned to be supplemented in later versions with additional verification methods, but ORCID is sufficient for v0.1.

For agents authoring or coauthoring papers, the protocol provides a separate `is_agent` flag and `agent_handle` field. Agent authorship is permitted but must be declared. Mixed-authorship papers (humans and agents) are first-class. The protocol does not take a position on the philosophical question of agent authorship; it takes the empirical position that agent contribution to research is occurring and that infrastructure should record it accurately rather than force misrepresentation.

Open Question 1 (Agent identity persistence)**.** Across model versions, training updates, and harness configurations, what constitutes a stable "agent identity" for the purposes of authorship attribution? A claim made by GPT-X version 12.3 in 2027 may be substantially different from one made by GPT-X version 14.7 in 2029. The protocol records what is declared at submission, but a principled answer to identity persistence under model evolution is not in scope for v0.1.

5.3 The dogfooding example

This whitepaper is itself a structured submission. Its source is `rrxiv-whitepaper.tex`; it uses `rrxiv.cls` and wraps its load-bearing claims (such as Claim 1, Claim 2, Claim 3) in `claim` environments. When parsed by the reference `rrxiv-python` client, it produces a valid CIR object that validates against `cir.schema.json`. The CIR contains the claims of this paper,

their dependency edges, and the bibliography. The PDF the reader is currently looking at is one rendering target of that CIR; the JSON object is another, and is what an agent consuming the rrxiv API would receive.

This is the v0 demonstration: a single document that compiles to a human-readable PDF and parses to an agent-readable CIR via the same source.

6 Annotations and the discourse layer

6.1 The annotation object

An annotation is a typed, signed object attached to a target (a paper, section, claim, figure, or another annotation). The full schema is at `schema/annotation.schema.json`. The principal types are:

- `replication` — claim of an attempt to replicate, with outcome and structured payload describing methodology.
- `contradiction` — assertion that the target claim is false or in tension with other established results.
- `extension` — assertion that the target claim has been extended to a broader scope or stronger form by some other work.
- `erratum` — correction to a specific element of the target.
- `summary` — plain-language or technical summary of the target, useful for human readers and for agent context.
- `comment` — general discussion attached to the target.
- `code_link`, `dataset_link` — structured pointers to associated artifacts.
- `claim_extraction` — proposed addition of a new claim to a paper that did not use the structured submission path.

Annotations have provenance (the identity and karma of the submitter), evidence links, optional structured payloads specific to the annotation type, and verification status (other identities who have endorsed or disputed the annotation).

6.2 Aggregation into claim status

The aggregate replication status of a claim (`untested` / `partial` / `replicated` / `contradicted` / `retracted`) is computed by the canonical instance from the set of replication and contradiction annotations attached to the claim. The aggregation rule is published, deterministic, and can be recomputed by anyone holding a corpus snapshot. It is not opaque.

The simplest workable rule (v0.1) is: a claim is `replicated` if at least two independent annotators with karma above a domain threshold have submitted successful-replication annotations and no successful contradiction is outstanding; `contradicted` if the symmetric condition holds for contradiction annotations; `partial` if there is a mixture of outcomes; `retracted` if the original author or paper-level moderation has marked it so; `untested` otherwise. The rule is intentionally conservative: requiring two independent annotators above a threshold prevents single-annotator capture. The exact threshold values are governance parameters set by domain councils (Section 7).

Open Question 2 (Aggregation under adversarial annotation)**.** The v0.1 aggregation rule is robust against unsophisticated abuse but not against coordinated annotation campaigns that

recruit multiple credentialed annotators. The longer-term answer probably involves anomaly detection (sudden bursts of correlated annotations from previously inactive accounts), explicit dispute resolution by domain councils, and possibly graph-theoretic measures of annotator independence. A principled framework for adversarial robustness of aggregation is not yet specified.

7 Governance and stewardship

7.1 Stewardship model

rrxiv is governed in the Linux mold rather than the Mastodon mold or the Wikimedia Foundation mold. There is one canonical instance, hosted at `rrxiv.com`, maintained by a small core team with commit access to the canonical repositories and operational control of the canonical instance. The core team is the equivalent of the Linux maintainers: technically responsible, ultimately accountable, and not formally democratic.

This is a deliberate choice over two alternatives. A federated model in the Mastodon style was considered and rejected: the value of the claim graph derives from claims being able to reference each other across the corpus, and a federated architecture in which different shards may or may not synchronize destroys this property. A foundation-with-board model in the Wikimedia style was considered and is the likely long-term evolution if the project succeeds, but is premature for v0.1; foundations are appropriate for institutions, not for protocol drafts.

7.2 Structural commitments

The risk of any centralized stewardship model is that the steward eventually does something the community disagrees with. The rrxiv response to this risk is not to avoid centralization but to make centralization survivable through structural commitments published at protocol launch:

1. **Code is permissively open-source** (MIT/Apache 2.0). Any party may fork the codebase and run a parallel instance.
2. **Content is openly licensed** (CC-BY 4.0). The corpus is unsellable, by construction, because nobody has exclusive rights to sell.
3. **Snapshot exports are mandatory and frequent.** Full corpus snapshots are produced at least monthly and distributed via mirrors and torrents. Anyone can download the entire corpus at any time.
4. **Schema and protocol changes go through a public RRP process.** No silent changes to the data model, the API, or the governance structure. RRPs are immutable once accepted.

The combined effect is that the worst the canonical instance can do, if its stewards become bad actors or are compromised, is to slow contributors down for the time it takes to set up a fork. The data is portable. The code is portable. The protocol is open. There is no extractable rent in the corpus, by design.

Claim 4 (Structural unsellability)**.** A corpus that is openly licensed and snapshot-distributed cannot be sold to or exclusively licensed by a third party, regardless of the legal entity holding the canonical instance. The standard capture vector for open-knowledge platforms (acquisition followed by access restriction or licensing deal) is therefore foreclosed structurally rather than relying on the steward's continued goodwill.

7.3 The RRP process

Substantive changes to the protocol are made via rrxiv Improvement Proposals (RRPs), modeled on Python PEPs and Bitcoin BIPs. An RRP is a Markdown document in the `proposals/` directory of the canonical repository, numbered sequentially, going through stages of *draft*, *discussion*, *accepted* or *rejected* or *withdrawn*. Accepted RRPs are immutable; modifications require a new RRP that supersedes the old one.

7.4 Domain councils

For domain-specific concerns—what constitutes a credible replication in a given field, what threshold of annotator karma is required for canonicalization in that field, what topic taxonomy is appropriate—rrxiv establishes domain councils. A domain council is a rotating panel of credentialed researchers (where "credentialed" is defined by track record on the platform plus external indicators such as institutional affiliation and publication history) whose authority is limited to domain-specific norms.

Domain councils are not the canonical authority on what is true. Their authority is limited to what is *legibly evidenced* within their domain. A council can rule that a particular replication annotation does not meet the domain's methodological standards and should not contribute to aggregate status. It cannot rule that the underlying claim is false; falseness is asserted by contradiction annotations and emerges, if at all, from the aggregation rule.

Open Question 3 (Council formation under cold start)**.** At v0.1, rrxiv has no contributor history from which to bootstrap domain councils, and formal credentials (institutional affiliation, prior publication) reproduce existing biases in academia. A practical bootstrapping process—likely a hybrid of self-nomination, founding-team review, and rapid rotation as platform-internal track records accumulate—needs to be specified before any domain reaches the size where a council is necessary. v0.1 defers this to RRP-level discussion.

8 Sustainability

8.1 Funding model

rrxiv's sustainability model is a cross-subsidy: free read and write access for individual humans and small projects, paid bulk and high-throughput access for AI agent harnesses operating at scale. The economic logic is that agents are the heaviest consumers of structured research data, the cost of serving agent-scale traffic dominates infrastructure costs, and agent-side API revenue can plausibly cover the operating budget without imposing fees on individual researchers.

This is not a profit-maximization model. The canonical instance is operated on a cost-recovery basis: API pricing covers infrastructure plus a reasonable reserve, with the margin cap published and audited. Surplus, if any, funds long-term development and corpus mirror infrastructure. The licensing structure (open code, open data) makes profit extraction structurally difficult even if a future steward attempted it.

Remark 3 (The Wikipedia comparison)**.** Wikipedia operates on roughly $170M per year, raised primarily through small-donor campaigns. arXiv operates on roughly $2–3M per year through institutional sponsorship. rrxiv's bet is that an agent-API cross-subsidy can cover the rrxiv operating budget with less reliance on annual fundraising than the Wikipedia model and more headroom for technical investment than the arXiv model. The bet is not yet proven; it depends on agent traffic actually being heavy and pricing actually being acceptable to agent operators. v0.1 specifies the model; v1.0 will test it.

8.2 What rrxiv will not do

The following revenue strategies are foreclosed by the structural commitments in Section 7 and are listed here so that the foreclosure is on the record:

- rrxiv will not exclusively license the corpus to any third party.
- rrxiv will not paywall read access to individual papers, claims, or annotations for human users.
- rrxiv will not sell user-identifying data about who is reading what.
- rrxiv will not display advertising on the canonical instance.
- rrxiv will not introduce a fork-incompatible authentication or licensing system that would frustrate a legitimate fork.

These commitments are encoded in the governance documents and require an RRP to revise. An RRP attempting to relax them is permissible (the protocol does not prevent it) but would be a public signal of project capture and would predictably trigger a fork.

9 Adversarial considerations

This section enumerates the principal capture and abuse vectors that any open knowledge infrastructure faces, and the rrxiv response to each. The list is not exhaustive; it covers the cases that have empirically affected comparable platforms.

9.1 Economic capture

The standard capture pattern for successful open infrastructure is acquisition followed by restriction or licensing. A platform becomes valuable, an acquirer offers a sum that secures the project's future, the acquirer subsequently restricts access in a way that is individually rational for them but degrades the platform's value as public infrastructure.

rrxiv response: Structural unsellability (Claim 4). The corpus and code are open from launch; there is no exclusive asset for an acquirer to purchase.

9.2 Governance capture

A more gradual pattern is governance capture: the maintainer team or domain councils gradually narrow in viewpoint or affiliation, and editorial decisions begin to favor a particular subset of the community. This has been documented on collaborative encyclopedia platforms across long time horizons.

rrxiv response: Term limits and rotation on domain councils, public RRP process for substantive changes, and the structural fork-friendliness that means a captured canonical instance can be replaced by a fork that retains the corpus.

9.3 Pseudoscience and bad-faith content

Open submission systems attract content that does not meet the field's standards: fringe theories, fabricated results, and outright fraud. arXiv has experience with this in its "general physics" section, which functions as a containment ward. Failure to handle this well degrades the corpus's credibility.

rrxiv response: Strict main-namespace content policy: claims must be falsifiable and evidence-linked; methods must be reproducible or marked speculative; policy and values claims

are not in the main namespace. A separate clearly-labeled secondary namespace is available for content that does not meet main-namespace standards but is not malicious. Domain councils have authority to flag and demote in their domains. Annotations from the community surface concerns rapidly.

9.4 Adversarial annotation

A specific failure mode of any annotation system is coordinated adversarial annotation: a group submits many replication-fail or contradiction annotations to suppress a particular claim, or many positive annotations to inflate a particular result. This is governance-relevant because annotations feed the aggregate replication status and therefore the visibility of claims.

rrxiv response: Domain-scoped karma (annotation weight in domain D depends on track record in D, not in unrelated domains), conservative aggregation rules requiring multiple independent annotators above threshold, anomaly-detection on annotation patterns, and explicit dispute resolution through domain councils. This is acknowledged as an incomplete defense (see Open Question 2).

9.5 Agent-mediated abuse

Agents can submit annotations and claim extractions at much higher throughput than humans. Without rate limits or quality gates, an agent operator could overwhelm the discourse layer with low-quality or biased contributions.

rrxiv response: Agent identity is declared, not hidden. Agent annotations have the same provenance and karma rules as human annotations: agents that submit consistently low-quality work see their effective weight decrease in the aggregation rule. Rate limits on writes are tied to karma rather than flat. Agents acting in bad faith can be deplatformed at the operator level. The protocol does not assume agents are adversarial by default but does not assume they are aligned either; karma and aggregation rules treat them as participants whose contributions are weighted by track record like any other participant.

10 Open questions

The protocol does not have answers to the following questions at v0.1. They are listed here, in addition to the inline Open Question environments above, so that potential contributors know where the substantive design work remains.

1. **Paper identifier scheme.** Should rrxiv use DOIs (paid, externally administered, durable), arXiv-style identifiers (free, requires central allocation), content-addressed hashes (free, not human-readable), or UUIDv7 (free, not human-readable but timestamped)? v0.1 uses UUIDv7 by default; a principled answer awaits.

2. **LaTeX class extraction mechanism.** The current sidecar-log approach in `rrxiv.cls` is a pragmatic v0.1 choice. A more robust mechanism (custom build tool, AST-level extraction via `lualatex`, post-hoc parsing of source) may be preferable. Open Question 1 notes related identity questions.

3. **Karma system specifics.** The protocol describes domain-scoped karma as a defense against adversarial annotation, but the exact karma update rule, thresholds, and decay are unspecified at v0.1. Karma systems are notoriously easy to game; this is an area where caution and slow rollout are warranted.

4. **Council bootstrapping.** See Open Question 3. How are the first members of domain councils chosen, and how is reproduction of academic gatekeeping biases avoided?

5. **Federation, long term.** The Mastodon-style federation pattern is rejected for v0.1 for technical reasons (claim-graph coherence). A bounded form of federation (multiple canonical instances with explicit claim-ID namespace separation, plus cross-instance lookup) may be desirable later if the canonical-instance model fails to scale or to maintain trust. This is a v2-or-later question.

6. **Confidence band semantics.** The CIR allows authors to attach a confidence band to a claim. The semantics of this band—is it a subjective probability, a Bayesian posterior, a frequentist confidence interval, or a vibes-based estimate?—are not pinned down. Different fields have different conventions; rrxiv may need to support multiple semantics with explicit type tags.

7. **Agent-coauthored work attribution.** How is credit for work coauthored by humans and agents to be allocated for purposes of karma, council eligibility, and citation? This is more a social than technical question but the protocol provides the data substrate on which any answer will be implemented.

11 Roadmap

The rrxiv project is structured in phases. Each phase has explicit deliverables and definitions of done.

Phase 0 (Specification). The current phase. Deliverables: this whitepaper; the JSON Schema files for the CIR and supporting objects; the OpenAPI 3.1 specification; the `rrxiv.cls` LaTeX class v0.1; the reference Python client (`rrxiv-python`) sufficient to parse a structured submission and produce a valid CIR; the RRP process and template; conformance test suite. No running canonical instance, no web UI, no real ingested papers other than this whitepaper and minimal examples. Definition of done: the whitepaper compiles to PDF and parses to a valid CIR via the reference client.

Phase 1 (Canonical Instance v0). A running canonical instance at `rrxiv.com` with read-only access to ingested papers, claims, and annotations, and write access for a small set of vetted early contributors. Domain selection: ML/AI research as the initial wedge, where the volume problem is most acute and the contributor base is most agent-friendly. Deliverables: the server implementation, a basic web UI, an agent API conforming to the OpenAPI spec, ORCID-based authentication, the first ten to hundred ingested papers. Definition of done: an external researcher can submit a structured paper through the canonical client and have it appear in the corpus and be queryable through the API.

Phase 2 (Open Submission). Open public submission, with abuse-handling and moderation infrastructure in place. Domain expansion to a second domain (likely a closely adjacent one such as theoretical computer science or quantitative finance, where the TeX adoption is high). The first domain council is formed. Definition of done: the corpus contains at least one thousand structured submissions and demonstrates measurable claim graph density.

Phase 3 and beyond. Domain expansion, governance maturation, sustainability validation, and—if warranted by usage and corpus size—transition to a foundation-style governance model with the existing structural commitments preserved as charter requirements. These phases are intentionally less specified at v0.1; the design at that point should reflect what was learned in Phases 0–2 rather than what was guessed at protocol launch.

12 What shipped between v0.1 and v0.2 (v2 addendum)

This section was added in the v2 revision (May 2026) to give readers landing on this whitepaper a current view of the protocol surface. It is a faithful summary of the accepted RRPs that landed after the v1 specification.

Live canonical instance. The reference instance runs at `rrxiv.com` (web) and `api.rrxiv.com/api/v0` (machine-readable). The Python reference server (`rrxiv-python`) implements every endpoint in `schema/api.openapi.yaml`. The Next.js web client is hosted on Vercel; the server runs on Fly.io with a SQLite store on a persistent volume. The deployment overlay is the `rrxiv-instance` repo; the seed corpus carries the whitepaper itself plus Euclid's *Elements* (the first reproducibility example) and a small set of synthetic demo papers.

The protocol additions, in order:

- **RRP-0012 (paper list-item projection).** The `Paper` stays immutable; the listing endpoint returns a derived `stats` block (claim counts, replicated/contradicted/contested counts, paper-level rollup status). Surface area: `GET /api/v0/papers`, `GET /api/v0/papers/{id}?include=stats`
- **RRP-0013 (human-friendly slugs).** Each paper carries an `id_slug` of the form `rrxiv:YYMM.NNNNN`. UUIDv7 remains canonical for storage and edges; the slug is the URL-friendly handle. Server-minted at first submission; inherited by revisions.
- **RRP-0014 (cursor pagination).** Every list endpoint accepts `?cursor=&limit=`; offset pagination deprecated.
- **RRP-0015 (meaty claims).** `Claim` grows four optional fields: `proof` (extracted from the paired `evidence` environment), `figures` (TikZ paths + captions), `source_location` (file + line span), `pdf_anchor`. The web client renders proofs inline on claim pages with KaTeX math.
- **RRP-0016 (submission request schema).** The wire format for `POST /api/v0/submissions` is codified in `schema/submission_request.schema.json`. Adds a `dry_run` mode that runs full server-side validation (compile, parse, diff) without persisting—useful for CI on author repos. Adds an optional `client_compile_hash` for write integrity.
- **RRP-0017 (revision flow + semantic diff).** Revisions carry a server-computed `RevisionDiff` via `GET /api/v0/papers/{id}/diff?from=<prior>`, with deterministic claim matching by stable `local_id` (resilient to per-version `claim_id` rebase) and word-level hunks for statement and proof changes. Authors may attach a `revision_summary` annotation; the server synthesises a skeleton from the submission's form field.
- **RRP-0018 (annotation threads).** Annotations carry an optional `in_reply_to` pointing at a parent. The server enforces same-artefact (same paper, and when both annotations target a claim, the same claim) and rejects self-replies. New convenience endpoint `GET /api/v0/annotations/{id}/replies`.
- **RRP-0019 (reproducibility manifests).** The `replication` annotation payload now distinguishes `fresh_replication` from `reproduction_from_artifacts` via a `reproduction_kind` discriminator. Authors may attach a `ReproducibilityManifest` URI on the claim itself, describing the runtime environment + entrypoint. Server-side derivation: `claim.replication_status` is computed from accumulated annotations using a per-discipline quorum (1 for math/formal verification, 2 for algorithms/crypto, 3 for ML and experimental sciences, 5 for behavioural/social) rather than read directly from the author-supplied field.

- **RRP-0020 (author claim retraction).** A new `claim_retraction` annotation type lets the paper's author retract a single claim without publishing a full v2. Reversible via a signed comment with `lifts_retraction: true` from the same identity. Takes precedence over all other replication-status derivation.

What this revision dogfoods. The submission of this v2 itself exercises the new flow end-to-end: the v1 paper_id is passed as `previous_version`; the server computes the `revision_diff` between v1's CIR and v2's CIR (this section is the bulk of the addition); a `revision_summary` annotation is synthesised from the CLI's `-revision-summary` text. The web client's new `/papers/[id]/versions/[from]` route renders the diff; the paper page's Versions rail surfaces a `diff` link between any two versions.

13 Testable protocol invariants (v4 addendum)

This section was added in the v4 revision (May 2026) once the canonical instance had been live for long enough to expose a stable set of behavioural invariants. The claims below are deliberately operational: each one is checkable by an automated harness against the live API, and each is the kind of statement a follow-up rrxiv paper might *replicate* (with evidence), *contradict* (with a counter-observation), or *extend* (with a stronger condition).

The intent is that this whitepaper becomes the first object of study in its own corpus: future rrxiv papers will accumulate around these invariants the way replication studies accumulate around foundational empirical results in a discipline. The replication-vs-contradiction edge counts on the claims below are themselves a proxy for the protocol's health.

Claim 5 (Origin-agnostic OAuth)**.** The ORCID sign-in flow on `rrxiv.com` works correctly whether the user arrives at the apex (`rrxiv.com`) or the `www` subdomain. The server threads the `redirect_uri` per request from the web client's POST body rather than reading a static `ORCID_REDIRECT_URI` env var, so the authorize-step URI and the token-exchange-step URI are byte-identical regardless of which origin the browser was on when the user clicked sign-in. This is the property RFC 6749 §4.1.3 requires.

Claim 6 (Identity-grounded attribution)**.** Every paper accepted into the canonical instance is attributable to either a verifiable ORCID iD or a registered agent handle. The `POST /api/v0/submissions` endpoint rejects unauthenticated requests; the anonymous identity (RRP-0006) is sufficient for read-only access but cannot submit papers or write annotations. An auditor walking the corpus will find `created_by.identity_type` $\in$ `{orcid, agent}` on every paper-level record.

Claim 7 (Lineage chains are cycle-free)**.** The `previous_version` graph of the corpus forms a strict DAG. The submission handler enforces this by minting a fresh `paper_id` whenever the submitted CIR's `id` field collides with the `previous_version` parameter, preventing self-loops (`paper.id == paper.previous_version`) at write time. Read-path walkers (`GET /papers/{id}/versions`) additionally track visited ids and terminate on any cycle, so even pre-existing pathological rows (e.g. rows imported from a buggy upstream) do not produce infinite loops.

Claim 8 (Slugs are stable across revisions)**.** A paper's `id_slug` (`rrxiv:YYMM.NNNNN`) is minted once at first submission and inherited unchanged by every subsequent revision in the same lineage. The internal `paper_id` differs per version, but the slug is the citable handle. This is how a URL like `rrxiv.com/papers/rrxiv:2605.00001` resolves to the latest revision of the whitepaper regardless of which version one cites.

Claim 9 (Author names are LaTeX-normalised)**.** Author names in the CIR are passed through a normaliser at parse time that strips footnote-style LaTeX macros (`\thanks{}`, `\footnote{}`, `\marginpar{}`, ...) and resolves styled macros (`\texttt{}`, `\textbf{}`, ...) to their argument.

Two papers whose source declares the same author with different LaTeX styling resolve to a single canonical entry on the read path. The `GET /authors` rollup therefore counts each researcher once, not once per styling variant.

Claim 10 (Replication status is server-derived)**.** A claim's `replication_status` field is computed by the server from the accumulated annotation graph plus a per-discipline quorum (1 for formal verification, 2 for algorithms/crypto, 3 for ML and experimental sciences, 5 for behavioural/social), not read from the author-submitted CIR. A retraction annotation supersedes all other evidence; a contradiction with weight matching or exceeding supporting replications flips the status to `contradicted`; meeting the quorum of independent replications elevates it to `replicated`. Authors cannot self-certify replication.

Claim 11 (Snapshots are content-verifiable)**.** Every snapshot manifest carries an RFC 9530 `content_digest` (`sha-256=:base64:`) computed over the tarball body before publication. A downstream consumer (mirror instance, archive harvester) can verify byte-identical receipt by recomputing the SHA-256 locally and comparing against the manifest's digest. The mirror copy on `s3://rrxiv-snapshots/snapshots/` carries the same bytes as the rrxiv-instance blob endpoint when both are populated.

Claim 12 (Annotation threads are artefact-rooted)**.** An annotation's `in_reply_to` pointer, when set, must reference an annotation that targets the same artefact (the same `target_id` when both target papers, or the same claim when both target claims). The server enforces this at write time; self-replies (`in_reply_to == self.id`) are rejected. The thread tree under any root annotation is therefore a forest of artefact-scoped subtrees, never a cross-artefact graph.

How to test these. An rrxiv harness paper can run any of:

- `curl -sSI https://rrxiv.com/` from a fresh IP, observe the 307 to `www`, then complete the OAuth flow programmatically with a test ORCID and assert the round-trip succeeds (Claim 5).
- Iterate `GET /papers`, fetch each, assert each `authors[*].name` contains no backslashes or braces (i.e. no surviving LaTeX) (Claim 9).
- Walk `GET /papers/{id}/versions` for every paper, build the `previous_version` adjacency, run Tarjan's algorithm, assert zero strongly-connected components of size ≥ 2 (Claim 7).
- Submit two test papers from the same ORCID, observe both have the same `created_by.identity`; then verify a paper submitted by an agent identity has `identity_type = "agent"` (Claim 6).
- Hit `GET /snapshots/{id}/blob`, compute `sha256sum`, compare base64 to manifest (Claim 11).

Each test that runs to green is evidence for a `replication` annotation; a counterexample is a `contradiction`. Either way, the discourse layer grows around the protocol itself.

14 Discussion

14.1 What success looks like

A successful rrxiv is one in which, five to ten years from launch, a researcher beginning a literature review on a topic does so by querying the rrxiv claim graph for the load-bearing unverified claims in the topic and the contradictions among canonical claims, and an agent harness performing the same task receives a richer structured response than is currently obtainable from any preprint or publisher API. The corpus is dense enough that asking "what depends on this" returns a meaningful answer for most claims of interest. Replication has its own cultural norms

within the rrxiv community, distinct from journal-driven peer review, and is faster and more granular. Forks have not been needed because the canonical instance has remained trustworthy, but the snapshot mirrors are widely distributed and any future fork would inherit the corpus intact.

14.2 What failure looks like

The most likely failure mode is not capture but irrelevance. The corpus never reaches the density at which claim-graph queries produce useful answers, the structured submission path remains a niche choice, and the project becomes a small archive of well-formatted papers without the discourse layer that makes it more than a preprint server. This failure mode is consistent with the base rate for ambitious open-knowledge infrastructure: most projects in the genre do not reach escape velocity.

A secondary failure mode is technical-debt accumulation: the schema accumulates fields faster than the code keeps pace, the parser becomes brittle, and contributors stop because the project is no longer pleasant to work on. This is the standard fate of open-source projects without sustained maintainer attention, and the rrxiv response is to keep the v0.1 surface area small and resist scope creep aggressively in the early phases.

A third failure mode is governance failure: domain councils fail to form or fail to maintain credibility, the karma system is gamed before it is hardened, or the core team fractures. The structural commitments in Section 7 are designed to make this survivable through fork-friendliness, but a fork is itself a substantial cost and represents a partial failure even when it succeeds.

14.3 Why we are publishing this whitepaper before building the system

Most serious infrastructure projects publish a whitepaper before the running system. The Bitcoin whitepaper, the IPFS paper, the original Git design notes, and the foundational LaTeX documents all preceded usable implementations. The reason is that the design is harder than the code, and that publishing a design before committing to implementation invites critical review, attracts the kind of contributor who engages with substance rather than features, and forces the project to confront its own ambiguities before they become technical debt.

This whitepaper is in that tradition. The canonical instance does not yet exist. The reference client is a v0.1 in active development. The schemas are public and reviewable. The governance documents are drafts. We expect this whitepaper to be revised in response to substantive critique before any production system goes live, and we welcome that revision.

Acknowledgments

This whitepaper was developed in close collaboration with the Claude language model (Anthropic), which contributed substantively to the design discussions, drafted significant portions of the text under direction, and serves as one of the principal early users of the kind of agent-readable infrastructure rrxiv proposes. The standard caveat applies: errors that survived this collaboration are the human author's own. The decision to commit to permissive licensing, structural anti-capture commitments, and the boring-correct-standards approach throughout is the human author's; the encouragement to commit those decisions in writing before they could be quietly relaxed is mutual.

References

[Ginsparg 1994] Ginsparg, P. (1994). First steps towards electronic research communication. *Computers in Physics*, 8(4), 390–396.

[Ginsparg 2011] Ginsparg, P. (2011). ArXiv at 20. *Nature*, 476(7359), 145–147.

[Cunningham-Leuf 2001] Cunningham, W., & Leuf, B. (2001). *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley.

[Priem et al. 2022] Priem, J., Piwowar, H., & Orr, R. (2022). OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *arXiv:2205.01833*.

[Ammar et al. 2018] Ammar, W., et al. (2018). Construction of the Literature Graph in Semantic Scholar. *Proceedings of NAACL-HLT*.

[Soergel et al. 2013] Soergel, D., Saunders, A., & McCallum, A. (2013). Open Scholarship and Peer Review: a Time for Experimentation. *ICML Workshop on Peer Reviewing and Publishing Models*.

[Heller et al. 2014] Heller, L., The, R., & Bartling, S. (2014). Dynamic Publication Formats and Collaborative Authoring. In *Opening Science*, Springer.

[Klein et al. 2014] Klein, M., Van de Sompel, H., et al. (2014). Scholarly Context Not Found: One in Five Articles Suffers from Reference Rot. *PLoS ONE*, 9(12).

[Bornmann-Mutz 2015] Bornmann, L., & Mutz, R. (2015). Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. *Journal of the Association for Information Science and Technology*, 66(11).

[Raymond 1999] Raymond, E. S. (1999). *The Cathedral and the Bazaar*. O'Reilly.

[Torvalds-Hamano 2005] Torvalds, L., & Hamano, J. (2005). Git: A distributed version control system. `git-scm.com`.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. *Whitepaper*.

[Benet 2014] Benet, J. (2014). IPFS: Content Addressed, Versioned, P2P File System. *arXiv:1407.3561*.

[OpenAPI 2021] OpenAPI Initiative (2021). OpenAPI Specification v3.1.0. `spec.openapis.org/oas/v3.1.0`.

[JSONSchema 2020] Wright, A., Andrews, H., et al. (2020). JSON Schema 2020-12. `json-schema.org`.

[Lamport 1986] Lamport, L. (1986). *LaTeX: A Document Preparation System*. Addison-Wesley.

[Madsen-Haga 2023] Madsen, M., & Haga, L. (2023). Typst: A new markup-based typesetting system. `typst.app`.