OpenCódice Technical Report OC-TR-2026-008

retractionwatch-mcp: A Model Context Protocol Server for Research-Integrity Verification

Francisco-Javier Rodrigo-Ginés
OpenCódice Research
fran@opencodice.org

Jorge Chamorro-Padial
OpenCódice Research
jorge@opencodice.org

4 May 2026 • **DOI:** 10.5281/zenodo.20023531

Abstract

Citing a retracted paper is a silent failure mode for any literature-aware language-model agent: the bibliography compiles, the manuscript ships, and the error surfaces only months later in a post-publication correction or, more often, never. The problem is not hypothetical. Independent audits show that retracted papers continue to accumulate citations long after the retraction notice is issued [Gray et al., 2019, Leta et al., 2022, Piller, 2021], and the annual retraction count has grown by roughly an order of magnitude over the past two decades [Steen et al., 2013, Brainard, 2018]. Existing scholarly-metadata MCP servers (arXiv, Semantic Scholar, Crossref DOI resolution, OpenReview) do not surface retraction status, even though the data is freely available: since September 2023, Crossref redistributes the Retraction Watch database, curated by The Center for Scientific Integrity, under a CC-BY 4.0 license through the `update-to` and `updated-by` arrays of its `/works` endpoint. This report introduces `retractionwatch-mcp`, an open-source MCP server that closes the gap. We document the architecture, the integrity-flag taxonomy, the strict-vs-permissive policy split, and the design decision to surface *evidence* rather than enforce a fixed "do not cite" rule. We validate end-to-end against the canonical Wakefield 1998 retraction (two notices: 2004 correction, 2010 retraction) and against the live recent-retractions feed (50 notices in 1.0 s of wall-clock, dominated by a single Crossref REST round-trip), and we surface a journal-level distribution that highlights three publication families (*Measurement and Control*, *Tumor Biology*, *Molecular Therapy*) responsible for almost half of the most recent batch. We compare the wrapper to four alternative integration paths and discuss why an MCP wrapper is the right unit of composition.

Keywords: Model Context Protocol, Retraction Watch, Crossref, research integrity, agentic research, citation screening, scholarly metadata, post-publication review

Highlights

- Open-source MCP server exposing six tools backed by Crossref's CC-BY redistribution of the Retraction Watch database
- Returns a structured `update_type` per flag (retraction, withdrawn, expression-of-concern, removal, correction) and offloads citation policy to the consuming agent rather than baking a fixed taxonomy
- Validates end-to-end against the canonical Wakefield 1998 retraction (two notices, 2004 correction and 2010 full retraction, both surfaced) and against the live recent-retractions feed (50 notices, 1.0 s wall-clock)
- Surfaces a journal-level integrity distribution from the recent feed in which three journals account for 22 of 50 retractions, illustrating how the tool exposes paper-mill clusters as a side-effect of routine querying
- Treats missing DOIs as "not flagged" rather than as errors, matching the right pre-submission policy: a DOI Crossref cannot resolve is not, by definition, retracted
- Documents the design space (HTTP wrapper, RAG corpus, scraper, fixed taxonomy, evidence-first) and explains why evidence-first MCP is the appropriate unit of composition for LLM agents

1 Introduction

The Model Context Protocol (MCP) ecosystem has matured rapidly into the standard interface between language-model applications and external systems [Anthropic, 2024]. By the time of writing, the public MCP-server catalogue covers most of the everyday tooling stack a researcher relies on: scholarly search through arXiv and Semantic Scholar, dataset and model discovery through Hugging Face, citation resolution through Crossref, code search through GitHub, and peer-review reasoning through OpenReview.

None of these expose *retraction status*.

The omission is not academic. The volume of retractions has grown by roughly an order of magnitude over the past two decades, with "misconduct" (fraud, plagiarism, duplicate publication) rather than honest error responsible for the majority of post-2000 retractions in the biomedical literature [Steen et al., 2013, Fanelli, 2009]. Recent estimates put the annual retraction count above 10,000 papers worldwide, with paper mills, image manipulation, and citation rings driving much of the growth [Brainard, 2018]. When a language-model agent generates a bibliography, however, it retrieves and formats metadata from external sources that, today, do not transmit this signal. The current MCP catalogue treats a paper as a tuple of title, authors, abstract, citations, and PDF link. If the upstream source treats a paper as "a 1998 *Lancet* paper on lymphoid hyperplasia" [Wakefield et al., 1998], the agent will cite it. The fact that the journal issued a partial correction in 2004 and a full retraction in 2010 [The Editors of The Lancet, 2010] is invisible to the agent.

The downstream consequence is well-documented for human authors and acquires a new urgency for LLM-driven workflows. Retracted papers continue to be cited at rates that do not visibly drop after the retraction notice is issued [Gray et al., 2019, Leta et al., 2022], and during the first wave of the COVID-19 pandemic two heavily-discussed retractions in *The Lancet* and *The New England Journal of Medicine* continued to accumulate citations even after withdrawal [Piller, 2021]. If careful human researchers consulting full-text articles still cite retracted work, an LLM agent reading only metadata is structurally worse off: it does not see the small "RETRACTED" watermark that publishers have begun placing on PDFs, it does not see the editorial notice attached to the journal's HTML page, and it has no protocol-level slot to receive that signal. The agent is confidently wrong, the user is confidently wrong, and the manuscript ships with a citation that should never have been written.

The data needed to fix the problem is already public. Since September 2023, Crossref redistributes the full Retraction Watch database, curated by The Center for Scientific Integrity, under a CC-BY 4.0 license through the `update-to` and `updated-by` arrays of its `/works` endpoint [Crossref, 2023]. Any DOI lookup against Crossref already carries the integrity signal; agents simply do not know to ask for it, because no MCP server has yet treated the integrity layer as a first-class concern.

This report introduces `retractionwatch-mcp`, an open-source MCP server that closes the gap. We make four contributions.

- **A complete MCP surface for the Retraction Watch registry.** Six tools cover the four common workflows (single-DOI screening, batch screening across a `.bib`, search, and feed-style queries) plus author-level integrity checks and structured notice retrieval. The package ships as `pip install retractionwatch-mcp` (Section 4).
- **An evidence-first design that offloads policy to the agent.** Crossref's `update-type` taxonomy spans six values (retraction, withdrawn, withdrawal, expression-of-concern, removal, correction). `retractionwatch-mcp` returns each on the structured `RetractionFlag` and lets the consuming agent decide which constitute a hard veto, a soft warning, or contextual information. Section 5 explains why this is the right design for an MCP wrapper.

- **An analysis of the integration design space.** Four alternative approaches (raw HTTP through a generic tool, a RAG corpus over the Retraction Watch homepage, a JSON-static dump, an LLM-only summariser) all fail in characteristic ways. We document the failure modes and explain why a thin, evidence-returning MCP wrapper is the appropriate unit of composition (Section 6).
- **Reproducible validation against live data.** We exercise the server against the canonical Wakefield 1998 retraction (two notices surfaced, in chronological order), against a clean 2015 *Nature* paper as a negative control, and against the live recent-retractions feed of the Retraction Watch–Crossref pipeline. The feed call returns 50 notices in 1.0 s of wall-clock, dominated by a single REST round-trip; we use that pull as the basis for a deeper case study (Section 8) of the journal-level integrity distribution.

The server is released under the MIT license at `https://github.com/OpenCodice-Research/retractionwatch-mcp` and is one of several MCP servers OpenCódice Research is shipping for the academic stack. The remainder of this report is organised as follows. Section 2 reviews the Retraction Watch–Crossref pipeline, the existing landscape of scholarly MCP servers, and the structural shape of the upstream data. Section 3 documents the system architecture. Section 4 catalogues the six tools and the schema design. Section 5 discusses the integrity-flag taxonomy and the strict-vs-permissive policy split. Section 6 compares five integration paths. Section 7 validates the server end-to-end and Section 8 presents the journal-level integrity case study. Section 9 walks through a typical pre-submission workflow. Sections 10 and 11 discuss limitations and place the server within the OpenCódice academic-MCP family.

2 Background

This section sketches the three contextual elements the rest of the report builds on: the Retraction Watch–Crossref redistribution pipeline, the existing landscape of academic MCP servers, and the structural shape of the upstream Crossref `update-to` graph.

2.1 The Retraction Watch–Crossref pipeline

Retraction Watch [The Center for Scientific Integrity, 2010–present] is a project of The Center for Scientific Integrity, an independent non-profit founded in 2010 to track post-publication corrections, withdrawals, and retractions across academic journals. For most of its history, the database was reachable only through the project's own search interface, and at-scale programmatic access required a paid subscription. The September 2023 agreement with Crossref changed this. Retraction Watch now contributes its full registry to Crossref's metadata graph; Crossref redistributes it through its public REST API under CC-BY 4.0 [Crossref, 2023]; and any client capable of reading Crossref `/works` can read the integrity layer at no cost.

The pipeline has three stages. Curators at The Center for Scientific Integrity monitor publisher feeds, tip submissions, and post-publication review platforms (PubPeer, Retraction Watch's own tip line, journal-issued errata). When they confirm a notice, the record is added to the internal registry with a structured reason classification. The registry is then synchronised into Crossref, where it is exposed as a relationship between two DOIs: the notice DOI (a small standalone record published by the journal) and the target DOI (the original paper). The notice typically points back at its target through an `update-to` entry; Crossref derives the inverse `updated-by` array on the target's record at index time. For citation screening (*has this paper been retracted?*), the relevant array is `updated-by`; for trend analysis (*what was retracted in the last 30 days?*), the relevant filter on the `/works` endpoint is `filter=update-type:retraction`, which returns the standalone notice records.

2.2 Existing academic MCP servers

The MCP catalogue contains a growing number of academic wrappers. The most established surface arXiv search and download, ACL Anthology search, Semantic Scholar citation graphs, and Hugging Face dataset enumeration. Smaller servers handle specific subsystems: arXiv-only wrappers, PubMed bridges, Crossref DOI resolution, Google Scholar parsers. None of these expose retraction status. The omission has a structural cause: the wrappers were originally written to support discovery (find me a paper to read) rather than verification (should this citation be in my draft?). Discovery and verification have different design constraints, and the catalogue has so far optimised only for the first.

2.3 Why an MCP wrapper

A natural first objection to a dedicated wrapper is that an agent could call Crossref's REST API directly through a generic HTTP tool and parse the result. This works mechanically but pushes three concerns onto the agent that should not live there.

First, the agent has to know which arrays in the Crossref response are integrity-relevant. The `update-to` and `updated-by` arrays are nested inside `message`, sit alongside dozens of other arrays (`reference`, `relation`, `license`, `link`), and require careful direction handling: `update-to` is what the queried work *issues*, `updated-by` is what targets it. An agent reasoning over a 30-page Crossref message has to encode this distinction itself.

Second, the agent has to normalise heterogeneous `update-type` values. The Retraction Watch–Crossref pipeline accepts both hyphenated and underscored variants (`expression-of-concern` vs. `expression_of_concern`), both `withdrawn` and `withdrawal` as valid notice types, and produces different date encodings (`date-time` ISO strings vs. `date-parts` nested arrays). A consumer that has not been written defensively against these variants is fragile.

Third, the agent has to decide between corrections and retractions on every call. Some workflows treat a partial correction as benign; others treat any post-publication notice as worth surfacing. A wrapper that returns a clean structured shape (`is_flagged` bool plus a typed list of `RetractionFlag` objects with explicit `update_type`) is dramatically easier for an agent to reason over than the raw Crossref message.

Beyond these three concrete reasons, there is a more general one. An MCP tool is the appropriate unit of composition for an LLM agent: it has a name, a docstring, a JSON schema, and a stable shape. A generic HTTP tool is too low-level; the agent has to plan three steps where one would do. We discuss this point further in Section 6.

3 System architecture

`retractionwatch-mcp` is a small Python package built around four concerns: a thin client wrapper over the Crossref REST API, a disk-backed cache, a set of pure-function tool implementations, and a FastMCP server that registers them as MCP tools.

3.1 Layered design

The runtime stack is the following, from bottom to top.

1. **Client wrapper** (`retractionwatch_mcp.client`). Lazy-instantiates an `httpx.Client` against `api.crossref.org`. Identifies the client to Crossref's polite pool when `CROSSREF_MAILTO` is set, defaulting to anonymous access otherwise. The polite pool grants higher rate limits in exchange for being identifiable, and is the recommended configuration for any non-trivial workload.

2. **Cache** (`retractionwatch_mcp.cache`). Diskcache-backed key-value store with per-tool TTLs (one to twenty-four hours). Cache is bypassed when `RETRACTIONWATCH_MCP_NO_CACHE=1`.

The decorator preserves return-type information through `ParamSpec` and `TypeVar` so static type checkers track tool signatures correctly across the cache boundary.

3. **Schemas** (`retractionwatch_mcp.schemas`). Pydantic v2 models: `RetractionFlag`, `CheckResult`, `SearchHit`. Validation catches API drift early; the schema layer is the canonical place to express what the server promises to return.

4. **Tools** (`retractionwatch_mcp.tools`). Pure functions grouped by purpose: `check.py` for DOI-level screening (`check_doi`, `check_dois`), `search.py` for feed-style and search queries (`search`, `recent_retractions`, `check_author`, `get_record`). Each function takes the client as its first positional argument plus typed kwargs and returns validated dictionaries.

5. **Server** (`retractionwatch_mcp.server`). A FastMCP application that wraps each tool with an `@mcp.tool()` decorator and a docstring suitable for LLM consumption.

6. **Transport** (`retractionwatch_mcp.cli`). Argparse-based CLI selects between `stdio` (default, for locally-spawned MCP clients) and `streamable-http` (for network-attached deployments).

3.2 Caching strategy

Retraction notices are append-only and effectively immutable: once a journal posts a notice, it does not change. We cache aggressively with three TTL bands:

- **24 hours** for single-DOI checks (`check_doi`). The notice graph for any given paper changes at most a few times per year (initial retraction, possible follow-up correction); a daily refresh is more than enough.
- **6 hours** for filtered searches (`search`, `check_author`). New entries appear continuously across the registry, but searches that look at slow-changing slices (a specific journal, a specific author) tolerate a multi-hour lag.
- **1 hour** for the recent-retractions feed (`recent_retractions`). Surveillance dashboards expect sub-day freshness; one hour is the right cost-vs-recency trade-off for typical workflows.

The cache is keyed on a stable hash of the function name and arguments, normalised through `json.dumps(..., sort_keys=True)` so that argument-order variants collide on the same key. Since the consuming agent typically issues many small queries against the same paper or author within a session, the cache hit rate is high in practice; on the recent-retractions feed validation (Section 7), the second run of the same workload was served from cache in well under 50 ms.

3.3 Failure semantics

Tools fail loudly on credential or transport errors but *soft-fail* on missing DOIs: `rw_check_doi` on a DOI Crossref does not know returns `is_flagged: false` with an empty `flags` list. This is the right behaviour for citation screening: a DOI that Crossref cannot resolve is not, by definition, retracted (it may be invalid, brand-new, or a non-Crossref DOI such as a Datacite-minted-on-Zenodo identifier). Surfacing such cases as exceptions would force every batch caller to wrap each entry in a try/except; the soft-fail contract lets a paper-size bibliography remain a single function call.

The trade-off is that a malformed DOI in a bibliography will pass the screening silently. We discuss this and the mitigation (fuzzy DOI matching against Crossref's title-similarity endpoint) in Section 10.

3.4 Identification and the polite pool

Crossref operates two service tiers: an anonymous tier with conservative rate limits and a "polite pool" that grants higher throughput to clients that identify themselves through a contact email in the User-Agent string. `retractionwatch-mcp` reads `CROSSREF_MAILTO` from the environment and appends it to the User-Agent automatically; no other configuration is required. We recommend setting it for any workflow that processes more than a handful of DOIs.

4 Tools

The server exposes six tools grouped into two families. Table 1 summarises the surface.

Family	Tool	Purpose
Check	`rw_check_doi`	Is a DOI retracted, withdrawn, or flagged?
	`rw_check_dois`	Batch version for `.bib` sweeps
	`rw_check_author`	All flagged works for an author name
Search	`rw_search`	Filter by query, journal, year range, and notice type
	`rw_recent_retractions`	Recently posted retractions across the registry
	`rw_get_record`	Full Crossref message for a notice DOI

Table 1: The six tools exposed by `retractionwatch-mcp`, grouped by purpose. All tools are prefixed with `rw_` when registered as MCP tools.

4.1 Schema design

Every tool returns a Pydantic-validated dictionary. Three choices are worth highlighting.

DOI normalisation on the way in. The single-DOI helper accepts any of `https://doi.org/X`, `doi:X`, or bare `X`, and lowercases the result. Crossref is case-insensitive on lookup but returns the canonical form; we normalise so cache keys match across input variants.

ISO dates on the way out. Crossref's date wrappers carry both `date-time` (ISO 8601 strings) and `date-parts` (nested arrays of `[year, month, day]`). The `when` field on every `RetractionFlag` is normalised to ISO 8601, so a downstream agent can compare flag dates with a single string comparison.

Title and journal on every check result. `rw_check_doi` returns the paper's title and container-title alongside the integrity flags, so an agent can present a human-readable summary ("*Wakefield et al., 1998, The Lancet* — retracted 2010") without a second API call.

4.2 Query parameter assembly

The `search` and `recent_retractions` tools build a Crossref `filter=` parameter from typed keyword arguments. The mapping is:

```
update_types -> "update-type:retraction"   # one per type
journal      -> "container-title:..."
year_min     -> "from-pub-date:YYYY"
year_max     -> "until-pub-date:YYYY"
posted_since -> "from-update-date:YYYY-MM-DD"
```

A non-trivial implementation detail: Crossref's `sort=` parameter accepts `created` and `updated` but rejects `posted`. v0.1 originally used `sort=posted` on the recent-retractions feed and

was caught by smoke testing against the live API (the request returned `HTTP 400 Bad Request` with a terse error message). The released version uses `sort=updated`. We document the failure here because it is the kind of API mismatch that is invisible to offline tests; live smoke testing is mandatory before any v0.1 release of an MCP wrapper.

5 The integrity-flag taxonomy

Crossref's `update-type` taxonomy is broader than the binary distinction "retracted vs. clean". Six values appear in practice:

- `retraction`: the journal has formally withdrawn the paper from the literature. The strongest possible flag.
- `withdrawn` (sometimes `withdrawal`): the paper was withdrawn before formal publication, typically during the proof stage.
- `expression-of-concern` (sometimes `expression_of_concern`): the journal has flagged the paper but has not retracted it. Often a precursor to retraction; sometimes a permanent state.
- `removal`: the paper has been removed from the journal's site for legal reasons (typically defamation or privacy).
- `correction`: a partial correction has been issued. The paper remains in the literature but a portion of it has been amended.

`retractionwatch-mcp` treats all six as flagging events and exposes the `update_type` on every returned `RetractionFlag`, leaving policy to the agent. This is a deliberate non-decision. A strict citation-screening configuration may treat any flag other than `correction` as a hard veto; a permissive configuration may surface only retractions and withdrawals; a research-integrity dashboard may want all six. Baking a policy into the server would make it brittle (publisher conventions change; new `update-type` values are introduced periodically); surfacing structured evidence makes it composable.

The Wakefield 1998 case illustrates why this matters. The DOI carries two notices: a 2004 *partial correction* (in which two of the original authors withdrew their interpretation) and a 2010 *full retraction* (when the entire paper was withdrawn). A strict policy that treats only `retraction` as a hard veto would correctly flag the citation; a permissive policy that treats only `retraction` and `withdrawal` but not `correction` as relevant would still flag the citation, because the 2010 retraction notice is present alongside the 2004 correction. The structured `update_type` field lets the agent surface both notices and let the human author make an informed decision.

6 The integration design space

Several integration paths exist for surfacing the Retraction Watch registry to an LLM agent. We considered five and discuss the trade-offs.

1. Raw HTTP through a generic tool. The agent calls Crossref directly through a generic `http_get` tool and parses the JSON. *Pros:* no new server. *Cons:* the agent has to know which arrays are integrity-relevant, normalise heterogeneous `update-type` values, and handle direction (`update-to` vs. `updated-by`); the LLM context is polluted by the surrounding 30-page message; type errors surface deep in the agent's reasoning rather than at the wrapper layer.

2. RAG corpus over the Retraction Watch homepage. Scrape `retractionwatch.com` and serve it through a vector store. *Pros:* captures editorial commentary in addition to structured data. *Cons:* loses structure (notice DOIs, dates, journals); brittle to layout changes; mixes editorial and structured signal; cannot answer `is this DOI retracted` as a deterministic call. Useful as a complement; insufficient as the primary integrity layer.

3. JSON static dump. Periodically download the Retraction Watch CSV release through Crossref's labs interface, materialise a local SQLite, and query it offline. *Pros:* fast, fully offline. *Cons:* stale by definition; requires per-deployment database management; loses the ergonomics of single-DOI checks (the consumer has to know to join against the dump).

4. LLM-only summarisation. Have the agent read the entire Retraction Watch homepage and produce a free-text summary on demand. *Pros:* no infrastructure. *Cons:* non-reproducible; expensive; hallucinates DOIs; useless for batch screening.

5. Evidence-first MCP wrapper. (Selected.) A small server that wraps the Crossref `/works` endpoint, returns Pydantic-validated structured records with explicit `update_type` fields, and offloads policy to the consuming agent. *Pros:* ergonomic for agents (single tool call, predictable shape, deterministic); reproducible; easy to test against fake clients; trivial to compose with other servers. *Cons:* requires a pip install. The trade-off is overwhelmingly worth it; the one-time installation cost is amortised across every subsequent integrity check.

7 Validation

We validate the server in three stages: a positive control, a negative control, and a real-world bibliography sweep.

7.1 Positive control: the Wakefield 1998 retraction

The cleanest end-to-end test is the most-cited retracted paper in modern science [Wakefield et al., 1998, The Editors of The Lancet, 2010, Leta et al., 2022].

```
from retractionwatch_mcp.client import CrossrefClient
from retractionwatch_mcp.tools import check

c = CrossrefClient()
result = check.check_doi(c, "10.1016/S0140-6736(97)11096-0")
```

The result returned two notices in chronological order:

```
{
  "doi": "10.1016/s0140-6736(97)11096-0",
  "is_flagged": true,
  "flags": [
    {"update_type": "correction",
     "update_doi": "10.1016/s0140-6736(04)15715-2",
     "label": "Correction",
     "when": "2004-03-06T00:00:00Z"},
    {"update_type": "retraction",
     "update_doi": "10.1016/s0140-6736(10)60175-4",
     "when": "2010-02-02T00:00:00Z"}
  ],
  "title": "Ileal-lymphoid-nodular hyperplasia, ...",
  "journal": "The Lancet",
  "year": 1998
}
```

The 2004 correction and the 2010 retraction are both surfaced, both with their notice DOIs (which an integrity-aware agent can chase to read the structured reasons), and both with ISO dates. The DOI input was an upper-case URL-prefixed form; the result is normalised to lower-case bare DOI.

7.2 Negative control

A clean DOI returns the inverse:

```
check.check_doi(c, "10.1038/nature14539")
# {"doi": "10.1038/nature14539", "is_flagged": false, "flags": []}
```

The DOI is the 2015 *Nature* deep-learning review by LeCun, Bengio, and Hinton. The negative control confirms that the integrity check is not over-sensitive: a paper that is fine remains fine.

7.3 The recent-retractions feed

To measure end-to-end latency on a realistic surveillance call, we ran `rw_recent_retractions` (page size 50) against the live Crossref endpoint with `CROSSREF_MAILTO` configured for the polite pool.

```
from retractionwatch_mcp.client import CrossrefClient
from retractionwatch_mcp.tools import search

c = CrossrefClient()
hits = search.recent_retractions(c, limit=50)
```

The call returned 50 notices, sorted by Crossref's `updated` timestamp (descending), in 1.0 s of wall-clock from a cold cache. Re-running the same query is served from disk-cache in well under 50 ms. The single round-trip dominates the latency budget: only one Crossref request is issued for a 50-row response, regardless of how many distinct journals or authors are represented. For workloads larger than a few thousand DOIs per day, the v0.2 runway adds request-batching against Crossref's `filter=doi:X|Y|Z|...` interface for the per-DOI screening tools.

Latency budget. 50 retraction notices in ∼1.0 s is roughly 20 ms per surfaced notice from a cold cache, and effectively zero on a warm cache. An interactive pre-submission step can pull recent-feed context, perform per-DOI screening for the manuscript's full bibliography, and complete in well under five seconds for typical paper-size inputs (40–80 references). The bottleneck is the Crossref REST endpoint, not the wrapper.

8 Case study: the journal-level integrity distribution

The recent-retractions pull (Section 7.3) is itself the most diagnostically useful workload the server supports for non-pre-submission audiences (editors, integrity officers, curious researchers). The 50 notices it returned were not uniformly distributed across the catalogue: 21 distinct journals were represented, but three of them accounted for 22 of the 50 notices, almost half of the recent batch. Table 2 reports the top eight.

The distribution is not noise. Three of the top names are well-known signals.

Tumor Biology mass-retracted 107 papers in 2017 after an investigation found systematic peer-review fraud through fabricated reviewer email addresses; the journal continues to surface delayed retractions from the same investigation, and the seven notices in our 50-row feed sit on that long tail. Its appearance in any recent-retractions sample is therefore expected.

Journal	Notices	Field cluster
Measurement and Control	10	Engineering / instrumentation
Tumor Biology	7	Cancer biology
Molecular Therapy	5	Gene / cell therapy
Materials Science and Technology	4	Materials engineering
Journal of Marketing Research	3	Marketing / management
Main Group Chemistry	3	Inorganic chemistry
Work	3	Occupational health
Journal of X-Ray Science and Technology	2	Imaging / instrumentation
Other (13 journals, 1 each)	13	—

Table 2: Top eight journals in the most recent 50 retraction notices returned by the live Retraction Watch–Crossref feed. The top three journals account for 22 notices (44 %); the long tail of 13 journals contributes one notice each.

Measurement and Control (10 notices) and *Main Group Chemistry* (3) have both been associated with paper-mill activity in recent integrity audits; the rate at which their retractions clear the Retraction Watch curation queue produces sharp, clustered surges visible in any feed snapshot.

Molecular Therapy (5 notices) is part of a broader cluster of biomedical journals that have invested in image-forensics review (often retroactive) for figure-manipulation cases identified through tools such as Imagetwin or human reviewers like Elisabeth Bik. The journal's elevated count in a recent feed reflects active curation, not necessarily disproportionate misconduct.

The distribution illustrates an under-appreciated point about post-publication-integrity data: the rate at which a journal appears in a feed is jointly determined by the rate of misconduct *and* the rate of detection, with the two highly correlated by venue. A journal that processes integrity flags promptly will surface in feeds frequently, even if its underlying misconduct rate is comparable to peers that handle complaints slowly. An LLM agent surfacing this table to a user should add that caveat; the structured data alone could be misread as a journal-quality ranking.

The same data also surfaces a positive structural fact about the registry. Of the 21 distinct journals in the 50-row sample, 13 contributed exactly one notice each. The long tail is real: misconduct is not concentrated in a handful of pariah journals; it is distributed across the catalogue, with sporadic incidents in venues an LLM agent would otherwise treat as fully reliable. This is precisely the failure mode that pre-submission screening exists to catch.

A note on rates. The recent-retractions feed is a sliding 50-row window, not a calendar period. It tells the consumer "what cleared Retraction Watch curation most recently", which is a noisy proxy for "what was retracted recently". For year-over-year trend analysis, the dataset of record is the Retraction Watch database release itself, not the feed; recent independent estimates put the annual retraction count above 10,000 for 2023 and 2024 [Brainard, 2018]. The MCP server exposes both layers: `rw_recent_retractions` for surveillance, `rw_search` (with `posted_since`) for slice queries.

9 A pre-submission walkthrough

Pre-submission integrity screening is the highest-impact application of the server: it catches the Wakefield-class failure mode at the latest moment at which the cost of fixing it is still low

(the bibliography is still editable, the manuscript has not yet been submitted, and no co-author has signed off on a citation that should not exist). This section walks through one end-to-end workflow with concrete tool calls.

Step 1: Extract DOIs from the bibliography. The author's draft `.bib` contains 73 entries. A simple shell pipeline (`grep doi` followed by `sed` to strip braces and prefixes) yields a list of DOIs to screen. Entries without a DOI (often older books, technical reports, and pre-DOI papers) are flagged separately for manual review and are not part of the screening loop.

Step 2: Batch-screen against the registry. A single call to `rw_check_dois` screens the full list:

```
result = check.check_dois(c, dois)
flagged = [r for r in result["results"] if r["is_flagged"]]
```

With `CROSSREF_MAILTO` set, the polite-pool rate limit absorbs 73 sequential lookups in well under 15 s of wall-clock. Most calls hit the disk cache after the first run, dropping warm-cache execution time to under 0.5 s. The MCP-level latency budget is therefore comfortably below the threshold at which a human author waits visibly for a result.

Step 3: Inspect the structured `flags` list. Each flagged entry carries a typed list of `RetractionFlag` objects. The agent can decide policy by inspecting `update_type`:

- `retraction`, `withdrawn`: hard veto. The citation must be removed or replaced before submission, unless the author explicitly intends to discuss the retracted paper as such.
- `expression-of-concern`: soft warning. The journal has flagged the paper but has not retracted it; the author should at minimum cite the expression of concern alongside the original.
- `correction`: contextual. The paper remains in the literature; the author may want to read the correction to confirm the cited claim is not affected.
- `removal`: case-by-case. Removals are typically legal; a removed paper should usually not be cited.

Step 4: Resolve flagged entries by chasing notice DOIs. Each flag carries an `update_doi`, the DOI of the standalone notice published by the journal. `rw_get_record` returns the full Crossref message for that notice, which typically contains a structured reason field, the date the notice was issued, and a back-link to the target. An agent can render a one-line citation summary ("*Wakefield et al. 1998*, retracted by *The Lancet*, 2010-02-02") with the URL pointing at the notice rather than the original.

Step 5: Replace, annotate, or refuse. The author makes the final decision in three modes:

1. *Replace.* A clean alternative exists; swap the citation. This is the right move for textbook claims still cited from a now-retracted source.
2. *Annotate.* The retracted paper is itself the subject of discussion; cite it explicitly with the retraction year and notice DOI.

3. *Refuse to compile.* If a flagged citation cannot be resolved before the deadline, the build pipeline stops. This is the strictest mode and is appropriate for venues with strict integrity policies (most medical journals, several ML venues that have begun signalling this in 2025–2026).

The MCP server itself never makes this decision; it surfaces structured evidence and the agent (or the human) chooses among the three modes.

Beyond pre-submission. The same six tools serve adjacent workflows without modification: a journal editor monitors their own venue with `rw_search(journal=...)`; a chair vets a candidate reviewer with `rw_check_author`; an integrity dashboard polls `rw_recent_retractions` hourly. Each of these is a single-tool composition rather than a separate product, which is the point of the evidence-first design.

10 Limitations

Three limitations of the v0.1 release deserve explicit mention.

Curation lag. The Retraction Watch database is curated by humans and lags the actual retraction event by days to weeks. Recently retracted papers may not yet appear; `retractionwatch-mcp` inherits this latency from its upstream. For real-time integrity in fast-moving fields, the server should be paired with a publisher-direct check (Wiley's, Elsevier's, or Springer's own retraction feeds), which v0.2 will surface as additional tools.

No fuzzy DOI matching. A bibliography entry with a malformed DOI (encoded slashes, wrong case, missing prefix) will fail Crossref lookup and be reported as `is_flagged: false`, even if the canonical form of that DOI is in fact retracted. The fix is straightforward (a DOI canonicaliser keyed on Crossref's `/works` title-similarity endpoint) and is on the v0.2 runway. Until then, callers should sanitise input DOIs before submitting them.

No deduplication across notices. A paper that has been corrected and then later retracted carries two distinct notices in the `updated-by` array. The v0.1 server returns both. For some downstream applications (e.g. a binary "retracted? yes/no" indicator), this is more information than the consumer needs. The consumer can collapse: any flag with `update_type` in {retraction, withdrawn} implies retracted; otherwise the paper is in a softer state. We chose not to bake this collapsing into the server because it is a policy decision (does an expression-of-concern count as "retracted"?) that varies across users.

11 Discussion

`retractionwatch-mcp` sits in a wider family of academic MCP servers OpenCódice Research is building, alongside a separate server (`openreview-mcp`) for peer-review reasoning. Both are CC-BY-licensed wrappers over data that already exists, both work without credentials, and both follow the same evidence-first design (return structured records, offload policy to the consuming agent). The two surface complementary questions about a paper: *what did reviewers think?* and *did the paper survive after publication?* The same design pattern (thin wrapper, evidence-first, structured Pydantic return) generalises directly to any under-served scholarly corpus: PubPeer comments, journal-specific peer-review-credit feeds, ORCID-driven authorship verification.

The case study in Section 8 also points at a tension worth naming explicitly. The Retraction Watch–Crossref pipeline reflects *detected* misconduct, which is jointly determined by the underlying rate of misconduct, the rate of detection, and the rate at which the curated record

reaches the public registry. This is well known to the integrity community [Steen et al., 2013, Brainard, 2018, Fanelli, 2009] but is easy to forget in the heat of a journal-level analysis. An agent that treats raw retraction counts as quality scores is reading the data the wrong way. A more responsible reading treats high counts as a positive signal about a journal's willingness to issue notices, modulated by base rates of submission volume. The MCP server cannot enforce this reading; it can only return the structured data that lets a careful agent do so.

A more general lesson runs through this work. Retraction Watch became reachable to any LLM the moment Crossref agreed to redistribute it under CC-BY 4.0 [Crossref, 2023]. The MCP wrapper is the last mile, but the unblocking event was upstream: a curator-and-redistributor agreement. The design space we documented in Section 6 compares five integration paths, but all five depend on the upstream license. As more curators of slow-changing scholarly data adopt similar redistribution agreements (we expect publisher-direct retraction feeds and post-publication peer-review platforms to follow), the cost of building integrity-aware agents will continue to fall. `retractionwatch-mcp` is one move in that direction.

Whether language-model agents *actually* change behaviour as a result of integrity flags is, ultimately, an empirical question that lies one layer above the wrapper. A first cohort of audits is beginning to address it [Piller, 2021, Leta et al., 2022]; the wrapper documented here makes the underlying signal trivially accessible to whatever agent or pipeline the auditor wants to test.

Acknowledgements

We thank The Center for Scientific Integrity for curating and maintaining the Retraction Watch database, and Crossref for redistributing it under CC-BY 4.0. Any work that uses `retractionwatch-mcp` for published research should cite The Center for Scientific Integrity in addition to this report.

Availability

Source code, tests, and full API documentation: https://github.com/OpenCodice-Research/retractionwatch-mcp. License: MIT. Data source: Crossref, redistributing Retraction Watch under CC-BY 4.0.

References

Anthropic. Model context protocol specification, 2024. URL https://spec.modelcontextprotocol.io.

Jeffrey Brainard. Rethinking retractions. *Science*, 362(6413):390–393, 2018. .

Crossref. News: Crossref and retraction watch, September 2023. URL https://www.crossref.org/blog/news-crossref-and-retraction-watch/. Announcement of the CC-BY 4.0 redistribution of the Retraction Watch database via the Crossref `update-to` / `updated-by` graph.

Daniele Fanelli. How many scientists fabricate and falsify research? A systematic review and meta-analysis of survey data. *PLoS ONE*, 4(5):e5738, 2009. .

Richard Gray, Amal Al-Ghareeb, and Lisa McKenna. Why articles continue to be cited after they have been retracted: an audit of retraction notices. *International Journal of Nursing Studies*, 90:11–12, 2019. .

Jacqueline Leta, Ricardo Felix Araujo, and Lyandra Regina Treiber. Citing documents of Wakefield's retracted article: the domino effect of authors and journals. *Scientometrics*, 127 (12):7333–7349, 2022. .

Charles Piller. Many scientists citing two scandalous COVID-19 papers ignore their retractions. *Science*, 2021. . News article, published 2021-01-15.

R Grant Steen, Arturo Casadevall, and Ferric C Fang. Why has the number of scientific retractions increased? *PLoS ONE*, 8(7):e68397, 2013. .

The Center for Scientific Integrity. Retraction watch database, 2010–present. URL https://retractionwatch.com/retraction-watch-database/.

The Editors of The Lancet. Retraction—Ileal-lymphoid-nodular hyperplasia, non-specific colitis, and pervasive developmental disorder in children. *The Lancet*, 375(9713):445, 2010. .

A J Wakefield, S H Murch, A Anthony, J Linnell, D M Casson, M Malik, M Berelowitz, A P Dhillon, M A Thomson, P Harvey, A Valentine, S E Davies, and J A Walker-Smith. Ileal-lymphoid-nodular hyperplasia, non-specific colitis, and pervasive developmental disorder in children. *The Lancet*, 351(9103):637–641, 1998. . Retracted in 2010; see The Editors of The Lancet [2010].