Vault Ingest & Sentinel Design – Working Recap

This note captures the agreed design for **vault ingest, writeback safety, and sentinel handling**. Treat it as an operational contract, not a sketch.

---

1. Overall ingest flow

```
JS (editor / Obsidian)
  → user marks file with sentinel (line 1)
  → Python parser reads files
    → emits NDJSON → asc ingest
      → on success emits NDJSON (batch slug + file list)
        → streamed to git safety script

On any failure: pipe breaks, nothing downstream runs.
```

Key property: - **No partial truth**. Either the whole chain succeeds or nothing happens.

---

2. Role of the sentinel

The sentinel is **not permanent** and **not decorative**.

It is a **transactional safety lock** with two simultaneous roles:

1. **Machine permission**
2. Signals that overwrite is allowed
3. Python must refuse to overwrite without it
4. **Human warning**
5. "Do not edit this file — it is volatile and may be overwritten"
6. Keeps the editor's brain engaged

Once writeback completes successfully, the file returns to normal human control.

---

3. Sentinel placement (critical invariant)

The sentinel must be the first line of the file.

Why this matters: - Structural certainty: no searching, no heuristics - Binary rule: line 1 == sentinel or not - Deliberately breaks YAML frontmatter and tooling - Visually unmistakable in the editor

This is a *feature*, not a side effect.

---

4. Sentinel lifecycle

1. Human (via JS tooling) inserts sentinel as line 1
2. Python ingest treats sentinel as permission to proceed
3. Python writeback:
4. writes new content
5. **removes sentinel as part of the same operation**
6. File is safe to edit again

Important: - Sentinel is **not removed on ingest** - Sentinel is **not removed on failure** - Sentinel is **only removed if Python also writes the file**

If anything crashes: - sentinel stays - overwrite risk remains explicit

---

5. Removal must be structural, not textual

Rules for removal: - Remove **exactly line 1**, no pattern matching - Assert line 1 is the sentinel before removal - Hard fail if line 1 is not the sentinel

No guessing. No partial matches. No forgiveness.

---

6. Git integration boundary

- Ingest emits NDJSON only on success
- Git safety script consumes NDJSON from stdin only
- Git runs **after** ingest, never before

Git concerns are strictly separated: - Ingest = semantic correctness - Sentinel = overwrite permission - Git = snapshotting successful outcomes

---

7. Core invariants (non-negotiable)

- Python must never overwrite a file without a sentinel
- Sentinel must never be auto-inserted by Python
- Sentinel removal and writeback are atomic
- Absence of sentinel = human ownership

- Failure anywhere leaves files untouched and visibly locked

---

8. Design intent (why this exists)

- No silent overwrites
- No permanent pipeline ownership
- No reliance on heuristics
- Human consent is explicit, visible, and revocable

This design optimises for tired humans, future debugging, and editorial ethics.