JCOOP v1.7

Journal-Centric Operational Provenance

Abstract

JCOOP is a journal-backed filesystem activity recording system designed to capture filesystem activity, preserve it as immutable evidence, and deterministically replay that activity under explicit policy control.

Unlike traditional backup systems, JCOOP does not attempt to preserve arbitrary file contents as its primary objective. Instead, it preserves a verifiable history of filesystem state transitions. This allows operators, auditors, engineers, and investigators to reconstruct filesystem state, validate historical integrity, selectively exclude events, and demonstrate the provenance of restored environments.

Version 1.7 establishes the project's core thesis:

> A filesystem history can be treated as evidence, verified independently, and selectively replayed to reconstruct a deterministic state.

---

1. Problem Statement

Modern systems answer one question well:

> What data do I have right now?

They answer a different question poorly:

> How did I get here?

Traditional backups preserve snapshots.

Version control systems preserve source code.

Log aggregation preserves messages.

None of these provide a deterministic, verifiable record of filesystem mutation history suitable for replay, auditing, or selective reconstruction.

JCOOP was created to address this gap.

2. What JCOOP Records

A common misconception is that JCOOP operates like a traditional backup system.

It does not.

Traditional backup systems typically preserve snapshots of data. At a given point in time, the system captures files, directories, and their contents, allowing those objects to be restored later.

JCOOP takes a fundamentally different approach.

Rather than recording the state of a filesystem at a particular moment, JCOOP records the filesystem operations that produced that state.

Conceptually, JCOOP observes filesystem activity such as:

```
mkdir /data

create /data/report.txt

write /data/report.txt

rename /data/report.txt → /data/report-final.txt

delete /data/report-final.txt
```

The journal therefore captures a history of filesystem mutations rather than a collection of snapshots.

This distinction is important.

A traditional backup answers:

> What did the filesystem look like at 10:00 PM?

JCOOP answers:

> What sequence of operations caused the filesystem to look the way it did at 10:00 PM?

Because JCOOP records operations rather than snapshots, individual events remain addressable after recording. A traditional backup can restore a filesystem as it existed at a point in time. JCOOP can replay the history that created that state

and selectively exclude individual operations during reconstruction. This capability forms the foundation of selective restore in Version 1.7.

For example:

```
CREATE alpha.txt
WRITE alpha.txt
WRITE alpha.txt
DELETE alpha.txt
```

can be replayed to reconstruct the resulting state.

The journal therefore functions as an operational history rather than a static archive.

This model provides several advantages:

- Reconstruction from recorded activity rather than snapshots.
- Deterministic replay of filesystem history.
- Selective exclusion of individual operations.
- Explicit provenance of reconstructed state.
- Independent verification of historical integrity.

The journal is therefore best understood as a record of filesystem behavior, not a backup of filesystem contents.

Technology	**Primary Question**
Backup	What data can I recover?
Snapshot	What did the system look like?
Version Control	What changed in source code?
Log Aggregation	What messages were emitted?
JCOOP	**What filesystem operations occurred?**

2.1 Why This Matters

Traditional backup preserves state. JCOOP preserves history.

Traditional audit determines what exists. JCOOP determines how it came to exist.

Traditional recovery is typically point-in-time based. JCOOP enables event-based replay.

3.Design Philosophy

JCOOP is built around several principles.

3.1 Journal First

The journal is the primary artifact.

Filesystem state is derived from the journal.

The journal itself is never derived from filesystem state.

This establishes a single source of truth.

3.2 Facts Over Inference

JCOOP records observed events.

It does not attempt to infer intent.

Examples:

Recorded:

- File created
- File modified
- File deleted

Not recorded:

- Why the change occurred
- Whether the change was correct
- Whether the change should be replayed

Intent remains a human concern.

3.3 Explicit Policy

Restoration behavior is controlled through policy.

Policy determines:

- Which events are replayed
- Which events are excluded

The journal remains unchanged.

Facts are preserved.

Interpretation is externalized.

3.4 Deterministic Results

Given:

- The same journal
- The same policy

JCOOP produces the same result.

No hidden state is permitted.

No external dependencies are consulted during replay.

---

4. Architecture

The JCOOP pipeline consists of five major stages.

```
Filesystem Activity
        ↓
Journal Capture
        ↓
Integrity Verification
        ↓
Policy Evaluation
```

```
        ↓
Deterministic Replay
```

Each stage has a narrowly defined responsibility.

---

5. Journal Model

Filesystem mutations are recorded as immutable journal entries.

Each entry includes:

- Sequence number
- Timestamp
- Operation type
- Path information
- Hash information
- Metadata
- Provenance information

Entries are stored in append-only journal segments.

5.1 Journal Replication

Journal segments may be replicated to independent storage locations; as the data is teed between the different locations, they can all be updated at the same time.

Examples include:

- Local storage

- Secondary systems

- Network shares

- Cloud storage

Because the journal is independent of the protected filesystem, evidence may survive even when the original system does not.

---

6. Integrity Model

Journal integrity is protected through two mechanisms.

6.1 Hash Chains, Not Blockchains

JCOOP uses hash chaining to make journal records tamper-evident. Each record references the hash of the previous record, creating a verifiable sequence in which modification of any entry invalidates subsequent entries.

This mechanism is similar to a blockchain in that both systems use cryptographic hashes to detect alteration of historical records. However, JCOOP is not a blockchain.

Blockchains are designed to establish consensus among multiple independent participants who may not trust one another. They typically require distributed validation, consensus protocols, and replicated ledgers.

JCOOP has a different objective. It assumes a defined trust boundary and focuses on preserving filesystem provenance within that boundary. There is no distributed consensus, mining, staking, or network-wide agreement process. The journal exists to provide verifiable evidence of filesystem activity, not to coordinate trust among unrelated parties.

Hash chaining in JCOOP therefore serves a narrower purpose: proving the integrity and ordering of recorded filesystem events so they can be independently verified and deterministically replayed.

Each record references the previous record’s hash.

This creates a tamper-evident chain.

Modification of any record invalidates subsequent records.

6.2 Segment Sealing

Journal records are stored in bounded collections known as journal segments.

A journal segment is an append-only sequence of journal entries that represents a contiguous portion of recorded filesystem history. Segments exist to make storage, verification, transport, and auditing manageable without requiring the entire journal to be treated as a single monolithic file.

While a segment is active, new records may be appended to it. Once a segment reaches a configured size, age, or operational boundary, it is finalized.

Finalization is the process of declaring that no additional records may be added to the segment. After finalization, the segment becomes immutable and is treated as a completed historical artifact.

As part of finalization, JCOOP computes a seal for the segment. The seal is a cryptographic summary derived from the segment's contents, including the ordered sequence of records and their integrity relationships. The seal serves as a compact representation of the finalized segment and establishes a trust boundary for subsequent verification.

During verification, JCOOP recomputes the segment seal and compares it to the recorded value. Any modification to the segment after finalization—including altered records, inserted records, removed records, or reordered records—causes seal verification to fail.

Because finalized segments are immutable and independently verifiable, they can be archived, transferred, or audited without requiring trust in the system that originally created them.

A seal therefore represents the trust boundary for the segment.

Any post-finalization modification becomes detectable during verification.

Journal segments are sealed when finalized.

A seal represents the trust boundary for the segment.

Any post-seal modification becomes detectable during verification.

---

7. Replay

Replay is the process by which JCOOP reconstructs filesystem history from a verified journal. Rather than relying on snapshots or inferred state, replay processes recorded events in sequence and applies them according to the journal's authoritative record.

Before any reconstruction occurs, replay validates the integrity of the journal. Hash chains, segment seals, and structural consistency checks are evaluated to ensure

that the history being replayed remains trustworthy. If verification fails, replay does not proceed beyond the point where trust can no longer be established.

Once verification succeeds, journal entries are processed in their recorded order. Each operation contributes to the reconstruction of filesystem state, allowing JCOOP to rebuild a historical view of the environment from the accumulated sequence of events. Because replay operates exclusively on journal data and explicit policy inputs, no hidden state influences the outcome.

Replay is deterministic by design. Given:

- The same journal
- The same policy

JCOOP produces the same result.

No hidden state is permitted.

No external dependencies are consulted during replay.

This determinism enables independent verification. Multiple parties can replay the same journal and compare results without requiring access to the original system that generated the records. Auditors, investigators, and operators can therefore validate reconstruction outcomes using the journal itself as the authoritative source.

Replay also serves as the foundation for higher-level capabilities within JCOOP, including selective restoration, historical analysis, integrity validation, and provenance verification. By treating the journal as evidence and replay as a reproducible reconstruction mechanism, JCOOP establishes a transparent and auditable path from recorded activity to reconstructed state.

Replay verifies journal integrity and reconstructs historical state transitions.

Replay is deterministic.

Given identical inputs, replay produces identical outputs.

Independent auditors can validate replay results without trusting the original recorder.

---

8. Selective Restore

Selective restore is the defining capability introduced in Version 1.7.

Traditional restore systems answer:

> Restore everything.

JCOOP can answer:

> Restore everything except specific events.

Example:

```
CREATE file.txt
MODIFY file.txt
DELETE file.txt
```

A policy may exclude:

```
DELETE file.txt
```

Result:

```
file.txt exists after restore
```

The journal remains unchanged.

The exclusion is documented and auditable.

---

9. Trust Boundaries

JCOOP explicitly defines trust boundaries.

Trusted:

- Verified journal segments
- Valid seal chains
- Explicit policies

Not trusted:

- Corrupted journals
- Tampered records
- Broken hash chains

Replay halts when trust can no longer be established.

This behavior is intentional.

Failure is preferable to uncertain reconstruction.

---

10. Restoration Semantics

Version 1.7 intentionally defines restoration as:

> Reconstruction of filesystem structure and state identifiers.

Version 1.7 does not guarantee reconstruction of original file contents.

This decision was made deliberately.

The objective of JCOOP is to preserve and replay state transitions rather than function as a traditional backup archive.

Future versions may expand content restoration capabilities, but they are not part of the Version 1.7 contract.

---

11. Validation Status

Version 1.7 has been validated through:

- Automated test suites
- Integrity verification tests
- Replay determinism demonstrations
- Tamper detection demonstrations
- Structural corruption demonstrations
- Divergence demonstrations
- Selective restore demonstrations
- Auditor verification demonstrations

The project currently maintains a fully passing automated test suite covering journal creation, replay, verification, policy enforcement, restore behavior, and trust-boundary enforcement.

12. Known Limits

Version 1.7 does not attempt to solve:

- Long-term archival storage
- Content-preserving backup replacement
- Distributed journal synchronization
- Cross-platform parity
- Intent inference
- Automated policy generation

These remain outside the scope of the current release.

---

13. Future Directions

Potential future work includes:

- Meta-validation of replay policies
- Human-readable consequence analysis
- Journal federation
- Distributed provenance chains
- Enhanced restoration strategies
- Policy simulation and impact analysis

These features would build upon the existing verified journal foundation rather than replace it.

---

14. Conclusion

Version 1.7 demonstrates that filesystem history can be captured, preserved, verified, and replayed as a trustworthy artifact.

JCOOP's central contribution is not backup, synchronization, or archival storage.

Its contribution is the treatment of filesystem activity as evidence.

By combining immutable journals, deterministic replay, explicit policy control, and verifiable trust boundaries, JCOOP provides a foundation for reconstruction, auditing, investigation, and selective restoration that extends beyond the capabilities of traditional snapshot-based approaches.

Version 1.7 establishes the project's core proposition:

A verified journal can serve as the authoritative record of filesystem history, and that history can be replayed deterministically under explicit human control.

Why This Matters

Traditional restore:

> Restore everything.

JCOOP:

> Restore everything except the ransomware delete operations.

Traditional audit:

> Determine what exists.

JCOOP:

> Determine how it came to exist.

Traditional verification:

> Trust the backup.

JCOOP:

> Verify the journal independently.