

RFC-005: Progressive Metadata Levels

Status: Implemented **Date:** January 2026 **Author:** Derrell Piper ddp@eludom.net **Implementation:** vault.scm (seal-commit, save-commit-metadata)

Abstract

This RFC specifies progressive metadata levels for vault commits, enabling configurable richness from minimal overhead to full preservation context. Three levels serve different use cases: minimal (fast iteration), catalog (discovery), and preserve (archival).

Motivation

Different commits deserve different metadata:

- **Quick fix:** Just record it happened
- **Feature release:** Add searchable keywords
- **Archival snapshot:** Capture full environment

Traditional VCS provides one-size-fits-all commit messages. Cyberspace provides progressive levels:

1. **Minimal** - Hash, timestamp, message (default)
 2. **Catalog** - Add subjects, keywords, descriptions (discovery)
 3. **Preserve** - Add environment, dependencies, git state (archival)
-

Specification

Level 1: Minimal (Default)

Every commit includes:

```
(commit-metadata
  (hash "abc123...")
  (timestamp 1767685100)
  (message "Fix typo in README"))
```

Usage:

```
(seal-commit "Fix typo in README")
```

This is the default. No extra flags needed. Git handles storage.

Level 2: Catalog

Adds discovery metadata:

```
(commit-metadata
  (hash "def456...")
  (timestamp 1767685200)
  (message "Implement user authentication")
  (catalog
    (subjects "authentication" "security")
    (keywords "login" "oauth" "jwt")
    (description "Added OAuth2 authentication flow with JWT tokens")))
```

Usage:

```
(seal-commit "Implement user authentication"
  catalog: #t
  subjects: '("authentication" "security")
  keywords: '("login" "oauth" "jwt")
  description: "Added OAuth2 authentication flow with JWT tokens")
```

Purpose: Enable search and categorization without full environmental capture.

Level 3: Preserve

Adds archival metadata:

```
(commit-metadata
  (hash "789abc...")
  (timestamp 1767685300)
  (message "Release v2.0.0")
  (preservation
    (environment
      (platform "darwin")
      (hostname "dev-machine")
      (chicken-version "5.x")
      (timestamp 1767685300))
    (dependencies
      (egg "srfi-1" "1.0")
      (egg "crypto-ffi" "0.1"))
    (git-state
      (branch "main")
      (remote "git@github.com:ddp/cyberspace.git"))))
```

Usage:

```
(seal-commit "Release v2.0.0"
  preserve: #t)
```

Purpose: Capture everything needed to understand and reproduce the commit environment.

Metadata Fields

Catalog Fields

Field	Type	Description
subjects	list	Library of Congress style subject headings
keywords	list	Free-form search terms
description	string	Extended description (beyond commit message)

Preservation Fields

Field	Type	Description
environment.platform	string	OS type (darwin, linux, etc.)
environment.hostname	string	Machine name
environment.chicken	string	Scheme implementation version
version		
environment.timestamp	integer	Unix epoch seconds
dependencies	list	Installed eggs/libraries
git-state.branch	string	Current branch
git-state.remote	string	Remote URL

Storage

Metadata stored in .vault/metadata/:

```
.vault/
  metadata/
    abc123.sex
    def456.sex
    789abc.sex
```

Filename is commit hash. Content is S-expression metadata.

Optional Git Tracking

```
(vault-config 'track-metadata #t)
```

When enabled, metadata files are staged for the next commit, creating a self-documenting archive.

Implementation

save-commit-metadata

```
(define (save-commit-metadata commit-hash
                               #!key message catalog subjects keywords description preserve)
  "Save optional metadata for a commit"
  (create-directory ".vault/metadata" #:t)

  (let ((metadata-file (sprintf ".vault/metadata/~a.sex" commit-hash)))

    ;; Build metadata structure
    (let ((metadata (list 'commit-metadata
                          (list 'hash commit-hash)
                          (list 'timestamp (current-seconds))
                          (list 'message message)))))

      ;; Add catalog if requested
      (when (or catalog subjects keywords description)
        (set! metadata (append metadata (list (build-catalog ...))))))

      ;; Add preservation if requested
      (when preserve
        (set! metadata (append metadata (list (build-preservation ...))))))

      ;; Write metadata file
      (with-output-to-file metadata-file
        (lambda ()
          (write metadata)
          (newline))))))
```

Environment Capture

```
(define (get-environment-snapshot)
  "Capture current build environment"
  `((platform ,(or (get-environment-variable "OSTYPE") "unknown"))
    (hostname ,(or (get-environment-variable "HOSTNAME") "unknown"))
    (chicken-version "5.x")
    (timestamp ,(current-seconds)))))

(define (get-dependencies-snapshot)
  "Capture current dependencies"
```

```

;; Could scan imports, check installed eggs
'())

(define (get-git-state-snapshot)
  "Capture git repository state"
  (let ((branch (with-input-from-pipe "git branch --show-current" read-line))
        (remote (with-input-from-pipe "git remote -v" read-line)))
    `((branch ,branch)
      (remote ,remote))))

```

Use Cases

Daily Development (Minimal)

```
(seal-commit "WIP: refactoring auth module")
(seal-commit "Fix off-by-one error")
(seal-commit "Update dependencies")
```

Fast, lightweight, no overhead.

Feature Completion (Catalog)

```
(seal-commit "Add two-factor authentication"
  catalog: #t
  subjects: '("authentication" "security" "2FA")
  keywords: '("totp" "authenticator" "login")
  description: "Implements TOTP-based 2FA per RFC 6238")
```

Enables later discovery: “find all commits about authentication”

Release Snapshot (Preserve)

```
(seal-commit "Release v2.0.0"
  preserve: #t)
```

Captures full environment for reproducibility and forensics.

Query Support

Future enhancement: query interface for metadata.

```

;; Find by subject
(vault-search subjects: '("authentication"))

;; Find by keyword
(vault-search keywords: '("oauth"))

```

```
;; Find by date range
(vault-search from: "2026-01-01" to: "2026-01-31")

;; Find with preservation data
(vault-search has-preservation: #t)
```

Security Considerations

Information Leakage

Preservation metadata captures: - Hostname (could reveal infrastructure) - Platform (could reveal vulnerabilities) - Dependencies (could reveal attack surface)

Recommendation: Use preserve level only for internal archives. Strip when publishing externally.

Metadata Integrity

Metadata files are not automatically signed. For tamper-evidence: 1. Enable track-metadata to include in commits 2. Use seal-release for cryptographic sealing 3. Audit trail captures metadata operations

Design Rationale

Why Not Always Full Metadata?

1. **Performance:** Environment capture adds latency
2. **Noise:** Not every commit needs forensic detail
3. **Privacy:** Some metadata reveals sensitive info
4. **Storage:** Full metadata increases repository size

Why S-expressions?

1. **Readable:** Human-inspectable without tools
2. **Parseable:** Machine-processable
3. **Extensible:** Add fields without schema changes
4. **Native:** Scheme can read/write directly

Why Separate Files?

1. **Git-friendly:** Small files diff well
2. **Query-friendly:** Can glob/grep metadata
3. **Optional:** Metadata doesn't bloat git objects
4. **Flexible:** Can delete without rewriting history

References

1. Dublin Core Metadata Initiative (DCMI)
 2. Library of Congress Subject Headings (LCSH)
 3. Software Heritage Archive Metadata
 4. Reproducible Builds Project
-

Changelog

- **2026-01-06** - Initial specification
-

Implementation Status: Complete **Test Status:** Passing (test-vault-metadata.scml) **Levels Supported:** Minimal, Catalog, Preserve