Custody Protocol — Implementation Roadmap & Library Stack

This document outlines the recommended dependency stack and phased implementation roadmap for building the Custody Protocol node. The focus is on simplicity, determinism, and auditability rather than maximum throughput optimization.

1. Recommended Library Stack

Core Cryptography:

• Ed25519: libsodium
• Secp256k1: libsecp256k1
• Hashing: BLAKE2b, BLAKE3, or SHA-256 (choose one consistently)

These libraries provide audited, constant-time implementations and should be wrapped behind a minimal
crypto abstraction layer inside the protocol codebase.

Storage:

• RocksDB (with prefix iteration enabled)
• Optional: Zstandard (zstd) compression for RocksDB values

All state values are SCALE-encoded and stored as raw bytes.

Networking / API:

• gRPC (if integrating with CometBFT or ABCI++)
OR
• Simple HTTP server (e.g., Boost.Beast or cpp-httplib)

Externally exposed APIs should prefer simple JSON responses for auditor clarity.

Configuration & Tooling:

• nlohmann/json (for genesis configuration and CLI tools)
• spdlog (structured logging)

Optional but Recommended:
• libFuzzer + sanitizers for deterministic testing
• CMake-based dependency management (vcpkg or Conan optional)

2. Implementation Roadmap

Phase 1 — Core Infrastructure

1. RocksDB wrapper (snapshot + WriteBatch abstraction)
2. Typed SCALE encode/decode layer
3. Deterministic key builder utilities
4. Nonce enforcement + signature verification layer

Phase 2 — Governance & Registry

5. Workspace & Vault state transitions
6. PolicySet creation + activation
7. Asset registry implementation
8. Destination registry implementation

Phase 3 — Custody Workflow

9. ProposeIntent handler (policy snapshot + validation)
10. ApproveIntent handler (threshold tracking)
11. Attestation subsystem (create/revoke + prefix scans)
12. ExecuteIntent validation logic (all gating checks)

Phase 4 — Node Integration

13. ABCI or consensus integration layer
14. Public query interface
15. Structured error codes
16. Deterministic event emission (optional)

Phase 5 — Hardening

17. Property tests for encode/decode roundtrips
18. Fuzz transaction handlers
19. Invariant testing (approval thresholds, timelock, attestation gating)
20. Security review and code audit preparation

3. Architectural Principles

• Deterministic execution only (no hidden state, no nondeterministic randomness)
• Explicit read/write sets per transaction
• No private keys stored inside node state
• Policy snapshotting to prevent historical rewrite
• Immutable audit artifacts (approvals + attestations)
• Minimal external dependencies for audit clarity