Covenant Protocol (C-Protocol) Specification

Version: 0.4.0-draft
Status: Working Draft
Date: September 10, 2025

Abstract

The Covenant Protocol (C-Protocol) is a universal, platform-agnostic meta-protocol for creating and exchanging verifiable claim tokens that bind digital assets to specific conditions. It leverages the Bitcoin ledger as a truth layer for on-demand settlement, while prioritizing privacy through off-chain negotiation. By combining **Party-Defined Proofs** from context-specific **Proof Authorities** with expressive, on-chain covenants, C-Protocol creates a secure bridge between real-world agreements and on-chain state transitions, enabling a peer-to-peer economy of high-trust, programmable value.

1. Introduction & Core Principles

1.1. Motivation

A significant gap exists in reliably connecting on-chain agreements to off-chain performance and conditions in a private, peer-to-peer manner. C-Protocol is designed to bridge this gap by providing a framework for creating verifiable, transferable claims on assets, where settlement is a final, optional step rather than a prerequisite for exchange.

1.2. Core Principles

1. **Party-Defined Proofs & Conditions:** The protocol's integrity rests on conditions and authorities chosen by the covenanting parties. This includes the ability to designate ephemeral, context-specific **Proof Authorities (PAs)**—such as an event organizer trusted to verify attendance—making the protocol highly adaptable to real-world, time-sensitive agreements.
2. **Immutable, On-Demand Settlement:** Covenants are Bitcoin Script programs. Their rules are enforced by the Bitcoin mining network, providing finality *when settlement is requested*. The protocol treats on-chain settlement as a final state transition, not a requirement for every interaction.
3. **Privacy Through Off-Chain Negotiation:** The exchange of claims and the verification of conditions happen primarily on private, peer-to-peer channels. Only final settlement transactions are broadcast to the public ledger, minimizing the on-chain footprint.
4. **Composable & Peer-to-Peer:** The protocol is designed for an ecosystem where any peer can be an issuer, holder, or verifier. It extends robust protocols like **CAT** to allow attested assets to be integrated into a wider ecosystem of decentralized applications.

2. Architecture & Implementation Patterns

C-Protocol is a layered architecture that supports multiple trust and custody models, allowing developers to choose the right trade-offs for their application.

- **Layer 1: Truth Layer (Bitcoin Protocol):** The ultimate settlement and state machine.
- **Layer 2: Logic Layer (Bitcoin Script / nPrint):** The enforcement engine for on-chain settlement.
- **Layer 3: Asset Layer (CAT Protocol):** The standard for representing assets governed by covenants.
- **Layer 4: Attestation & Custody Layer:** A flexible layer of entities and models for managing assets and verifying proofs.
- **Layer 5: Origination & Negotiation Layer:** Where actions occur and claims are exchanged peer-to-peer.

2.1. Custody & Redemption Models

Peers can implement C-Protocol using various models, ranked by decentralization:

1. **Simple Server-Mediated:** A single custodian holds assets in escrow.
2. **Federated Threshold Custody:** An M-of-N quorum of peers manages a threshold wallet.
3. **State/Payment Channels:** Parties update asset states off-chain with signed messages.
4. **Atomic Conditional Releases (Script-based):** Primitives like Adaptor Signatures or HTLCs tie asset release atomically to the revelation of a secret.

3. Core Components

3.1. Proof Authorities (PAs)

A Proof Authority is any entity—an individual, organization, automated sensor, or software program—that is designated by the covenanting parties to attest to the state of the real world for the purposes of their agreement. The protocol is unopinionated about who can be a PA; the trust is entirely defined by the participants.

- **Context-Specific:** A PA's authority is typically limited to a specific domain. For example, a shipping company's API may be a PA for "proof of delivery," but not for "proof of identity."
- **Ephemeral:** A PA's authority can be time-bound. For a beach cleanup, the organizer acts as a PA whose ability to issue valid attendance proofs is only active during the event. This allows for the creation of dynamic, temporary trust relationships that expire safely.

3.2. The Covenant Contract

A UTXO controlled by a Bitcoin Script program. The script defines the programmable conditions for spending the UTXO.

3.2.1. Expressive Conditions

Covenants can implement a rich predicate language for defining unlock conditions. This logic is enforced by the custody model or directly in Bitcoin Script. Examples include:

- holder_sig: Requires a valid signature from the claim's holder.
- reveal:hash(H): Requires the holder to reveal a secret pre-image x such that SHA256(x) == H.
- time_lock:T: The claim is only redeemable after a specific time or block height T.
- multi_sig:[pubA, pubB]: Requires signatures from multiple specified parties.
- pa_attested:verify_sig(PA_PubKey, proof_data): Requires a valid signature from a designated Proof Authority over specific proof data (e.g., attendance stats).

3.3. Verifiable Claims (Canonical Obligations)

The core transferable object in the C-Protocol ecosystem is the **Canonical Obligation**, a signed data structure representing an offer or a finalized receipt. This object is what peers trade.

```
{
  "type": "offer" | "receipt",
  "id": "unique_obligation_id",
  "issuer_pub": "public_key_of_issuer",
  "holder_pub": "public_key_of_current_holder",
  "token_info": { "chain": "BSV", "assetId": "...", "amount": 10000 },
  "condition_expr": "pa_attested:verify_sig(02a4..., 'cleanup_stats')",
  "expiry": 1757283600,
  "nonce": "random_bytes",
  "signatures": [ "signature_from_issuer" ]
}
```

This structure allows a claim on an asset to be securely and verifiably transferred between parties off-chain.

4. Protocol Flow: Reserve → Offer → Claim → Settle

1. **Reserve:** An **Issuer** locks an asset using one of the chosen custody models.
2. **Offer:** The Issuer creates a signed Canonical Obligation of type offer, specifying the conditions for redemption, including any required PA.
3. **Claim:** A **Holder** receives the offer. To claim it, they fulfill the condition_expr (e.g., by obtaining a signed attestation from the designated **Proof Authority** for their cleanup stats) and present this proof to the responsible custody peer(s).
4. **Settle:** The custody layer verifies the claim and the PA's attestation. Upon success, it either updates its internal off-chain ledger or, if requested, executes the final on-chain settlement.

5. Privacy Measures

- **Off-Chain First:** All negotiation, offers, and claims are exchanged on private, encrypted peer-to-peer channels.
- **On-Demand Settlement:** The public blockchain is only used for final settlement.
- **Ephemeral Identifiers & Address Rotation:** Peers should use short-lived identifiers and rotate addresses.
- **Batching & Merkle Anchoring:** For public verifiability, peers can periodically anchor a single Merkle root of many off-chain obligations to the blockchain.

6. Security Considerations

- **L1 Double-Spend Risk:** Each peer should run a lightweight **watcher** node that monitors the L1 backing UTXOs. If a backing asset is spent, all dependent off-chain obligations must be revoked.
- **Issuer Repudiation:** Holders can require offers to be co-signed by a trusted quorum or demand a non-custodial, script-based model.
- **Claim Replay Attacks:** Claims must be made single-use through the use of nonces.
- **Peer Collusion / Eclipse Attacks:** Clients should connect to a diverse set of independent peer nodes and watchers.