PRECOP: Programmable Sovereignty and Deterministic Transaction Derivation on Bitcoin Layer 1

The PRECOP Working Group

Version 0.1.0-alpha – April 15, 2026

Institutional Review Draft

Subject: Deterministic Covenants, Simplicity Execution, and Transaction Integrity

Abstract

Bitcoin's consensus layer enforces data integrity but remains agnostic to the semantic correctness of higher-order financial protocols. Current meta-protocols (Ordinals, BRC-20, Runes) rely on **Passive Observation**, where external indexers reconstruct state from raw on-chain data, creating a *Sovereignty Gap* between ownership and enforcement. This paper introduces **PRECOP** (Predictive Covenant Oracle Protocol), a deterministic canonical transaction derivation engine that eliminates the PSBT Trust Problem by ensuring that every signed transaction is the unique, mathematically necessary realization of a formally defined policy. PRECOP combines Simplicity's CMR-based program identity, Taproot's MAST isolation, and a fail-closed execution model to enforce state transitions *before* a signature is produced. The system operates within Bitcoin's existing consensus rules, achieving byte-for-byte equivalence with `rust-bitcoin` sighash derivation. We present the formal model of the **Template-Enforced UTXO State Machine (TUSM)**, the **Astrolabe Pattern** for state-address fusion, and the **Canonical Engine Architecture**, and contrast this approach with indexer-dependent frameworks such as the BOSS meta-protocol. This work establishes a new paradigm: moving from *verifying* transaction correctness to *deriving* transaction necessity.

This version operates in a **Secure Shell Mode** (Tier 2 enforcement only) pending upstream stabilization of Simplicity Bitcoin jets. The architecture guarantees that no signature is produced for a non-canonical transaction.

Preamble

The Crisis of Interpretive Consensus The emergence of higher-order protocols on Bitcoin has exposed a fundamental architectural rift. While Bitcoin's consensus rules guarantee the integrity of the ledger, they do not enforce the semantic validity of the data inscribed upon it. Systems such as Ordinals, BRC-20, and the broader BOSS (Bitcoin Operational Standard System) framework operate on a model of **Passive Observation**. In this paradigm, Bitcoin serves merely as a broadcast and timestamping medium; the "truth" of the system's state—be it token balances, loan collateral ratios, or exchange order books—is relegated to an *external consensus layer* comprised of indexers and off-chain validators.

This architecture introduces what we term the **Sovereignty Gap**: users possess the cryptographic keys to their funds but lack the native means to enforce the rules governing their expenditure. They are compelled to trust that their wallet interface, their chosen indexer, and the broader peer-group share a consistent interpretation of an ambiguous on-chain history. The BOSS lightpaper itself acknowledges this dependency, stating that indexers are "developed based on subjective opinions, which can compromise the reliability of protocol data."[1]

PRECOP is founded on the rejection of this interpretive model. We posit that for a financial system to be truly sovereign—where *code is law*—state transitions must be **enforced, not observed**. This work defines a shift from *Verification* (checking if something was done correctly after the fact) to *Canonical Derivation* (ensuring that only the correct thing can ever be signed). By anchoring Simplicity's formal semantics within a deterministic derivation engine, we move Bitcoin logic from the realm of "meta-data interpretation" into the realm of **Mathematical Necessity**.

[1] BOSS Lightpaper, Section "Challenges."

1 Introduction:

The Silence of Layer 1 and the Blindness of Layer 2

1.1 Expressivity vs. Enforcement: The Taproot Paradox

With the activation of Taproot (BIP-341/342) and the theoretical introduction of Simplicity, Bitcoin has achieved a state of near-infinite script expressivity. Complex financial logic—including **covenants**, vaults, and stateful contracts—can now be encoded within Bitcoin's transaction structure without bloating the blockchain. However, expressivity alone does not guarantee security. The Bitcoin network remains fundamentally *agnostic* to the economic meaning of the scripts it validates. It ensures that a signature is valid, but it does not ensure that the transaction conforms to the *intent* of the protocol designer or the **sovereign actor**.

This distinction has given rise to a parallel execution layer often referred to as **Meta-Protocols**. Systems like BOSS (Bitcoin Operational Standard System) exemplify this architecture. BOSS utilizes *Smart Inscriptions*—JSON payloads embedded within Ordinals envelopes—to define operational instructions for token transfers (BRC-20) and other stateful interactions. As described in its lightpaper, BOSS relies on a network of **Simplifiers** (indexers) to "ingest and digest" these inscriptions, updating a virtual state machine (*MetaState*) on the **enforcement layer**.[2]

1.2 The BOSS Antithesis: Trust in the Indexer

While BOSS provides a framework for innovation on Bitcoin, it does not resolve the fundamental trust asymmetry of meta-protocols. The BOSS lightpaper candidly acknowledges this limitation:

> "Trust in the accuracy and integrity of this data is crucial for ensuring proper protocol functioning. However, this trust is often questioned as it relies solely on indexers [...] developed based on subjective opinions, which can compromise the reliability of protocol data."

This is the **Indexer Trust Problem**: the validity of a state transition (e.g., a BRC-20 transfer) is contingent upon the software version and interpretation logic of a third-party observer, not on Bitcoin's consensus rules. A transaction that is *valid* on the Bitcoin network can be *invalid* according to a specific indexer, leading to state forks, double-spend vulnerabilities in the application layer, and a general erosion of sovereign control.

[2]BOSS Lightpaper, Section "Definitions of the components of the BOSS solution."

1.3 The Convergence of Divergent Dreams: From Account Model to Canonical UTXO

In his seminal 2013 whitepaper introducing Ethereum, Vitalik Buterin argued that Bitcoin's UTXO model was inherently limited in its capacity for complex financial logic, necessitating an account-based model for persistent state memory [6]. PRECOP resolves this 13-year divergence not by adding a new state layer, but by enforcing a **Active Derivation** over existing UTXOs.

We demonstrate that the expressivity sought by Buterin is achievable through client-side state reconstruction that is cryptographically bound to the script via Taproot. Where Ethereum relies on global gas-metered consensus, PRECOP relies on **Local Mathematical Necessity**: the UTXO becomes a state-bearing machine whose transitions are as rigid as the laws of physics.

PRECOP is architected in direct opposition to this model. We do not seek to improve the *indexing* of state; we seek to make the *signing* of a transaction contingent upon the cryptographic proof that the state transition is correct. This requires addressing a more immediate and dangerous vulnerability: the PSBT Trust Problem.

1.4 The PSBT Trust Problem: Signing in the Dark

Modern Bitcoin transaction workflows rely heavily on the **Partially Signed Bitcoin Transaction (PSBT)** standard (BIP-174). In a typical scenario:

1. A coordinating application (wallet, DApp interface) constructs a PSBT.
2. The PSBT is serialized and presented to a signing device (hardware wallet).
3. The signer parses the PSBT, displays human-readable fields (amounts, addresses), and, upon **sovereign actor** approval, produces a signature.

This process assumes that the PSBT accurately represents the **sovereign actor's** intent. However, this assumption fails catastrophically under advanced sighash configurations, particularly **SINGLE—ANYONECANPAY (0x83)**. In such modes, the signer's signature covers only a subset of the transaction's inputs and outputs. The remainder of the transaction is *mutable* from the signer's perspective.

We define the **PSBT Trust Problem** as follows:

> *A signer produces a valid cryptographic signature based on a local, surface-level inspection of a PSBT, without cryptographically verifying the full structural, economic, and state-machine integrity of the complete transaction.*

1.5 Attack Vectors Enabled by PSBT Blindness

The PSBT Trust Problem enables several concrete attack classes that are invisible to standard wallet interfaces:

- **Fee Siphoning:** An attacker modifies the PSBT to add additional inputs or inflate output values. Because the signer's signature does not cover these changes (due to `SIGHASH_SINGLE` or `ANYONECANPAY`), the transaction remains cryptographically valid. The excess value is silently redirected to miners as inflated fees, effectively draining the **sovereign actor's** UTXO without their awareness.
- **Output Injection:** An attacker appends a new output to the transaction. The **sovereign actor** signs, believing they are participating in a simple swap, but the transaction also executes a hidden transfer of funds to an attacker-controlled address.
- **Topology Manipulation:** The ordering of inputs and outputs is altered, violating the expected execution path of a covenant or state machine (e.g., shifting the index of a required change output).

These attacks exploit a critical gap: **the signer reacts to a provided structure instead of deriving the structure independently.**

1.6 PRECOP's Foundational Shift: Canonical Derivation

PRECOP eliminates the PSBT Trust Problem by introducing a **Canonical Derivation** model. In this architecture, the signing enclave does not *receive* a transaction to approve. It *reconstructs* the transaction from first principles based on a formal `ContractPolicy`. The process is governed by a strict invariant:

$$\text{Sighash}_{\text{PSBT}} \equiv \text{Sighash}_{\text{Derived\_Transaction}} \tag{1}$$

If any byte diverges—whether it be a single satoshi of fee, a sequence number, or a script byte—the equality fails, and the system terminates in a **Fail-Closed** state. No signature is ever produced for a non-canonical transaction. This transforms the signer from a passive validator into an **Active Enforcer of Mathematical Correctness**.

2 Theoretical Foundations: The Template-Enforced UTXO State Machine (TUSM)

In the standard Bitcoin model, an Unspent Transaction Output (UTXO) is a passive container defined by the tuple $(\text{value}, \text{script\_pubkey})$. While sufficient for simple payments, this abstraction lacks the necessary primitives for expressing *stateful systems*: memory of prior transitions, constraints on future transitions, and an internal representation of state. Meta-protocols such as BOSS circumvent this limitation by maintaining state off-chain, introducing the indexer dependency previously critiqued. PRECOP takes a fundamentally different approach: it transforms the UTXO into a **deterministic, state-constrained execution object**.

2.1 Formal Definition of TUSM

A PRECOP-governed UTXO is an instance of a **Template-Enforced UTXO State Machine (TUSM)**. At any block height t, the state of a TUSM instance is defined by the 5-tuple:

$$S_t = (id_t, \phi_t, C_t, D_t, h_t) \tag{2}$$

where:

- $id_t \in \{0,1\}^{128}$ is a unique identifier ensuring **domain isolation** between independent state machines. This prevents cross-contamination or replay of state across distinct contracts.
- $\phi_t \in \mathbb{N}$ (bounded) is the **lifecycle phase**. Common encodings include:

  $$\begin{aligned} \text{MINT} &= 3 && \text{(Collateral deposit / debt issuance)} \\ \text{REPAY} &= 4 && \text{(Debt repayment)} \\ \text{LIQUIDATE} &= 5 && \text{(Forced liquidation)} \\ \text{FREEZE} &= 7 && \text{(Terminal / halted state)} \end{aligned}$$

  This phase variable enforces a strict, deterministic control flow, eliminating ambiguous state transitions.
- $C_t \in \mathbb{N}$ is the **collateral amount** (in satoshis).
- $D_t \in \mathbb{N}$ is the **outstanding debt** (with fixed precision, typically 10^8 units).
- $h_t \in \mathbb{N}$ is the **block height** of the last valid transition. Its monotonic progression ($h_{t+1} > h_t$) provides a native replay protection mechanism.

2.2 The Deterministic Transition Function δ

The evolution of the state is governed by a deterministic transition function:

$$\delta : S \times \Sigma \to S' \tag{3}$$

Here, Σ denotes the **witness set**, which includes:

- The transaction inputs and outputs being evaluated.
- Simplicity witness data (e.g., Flare values for arithmetic constraints).
- Oracle inputs (if applicable) and block context (height, timelock).

A transition from S_t to S_{t+1} is valid **if and only if** the execution of a pre-committed Simplicity program yields a true result:

$$\text{BitMachine}(AST, \Sigma, S_t) = \text{true} \tag{4}$$

where AST is the Simplicity Abstract Syntax Tree uniquely identified by its Commitment Merkle Root (CMR). This condition ensures that the transition is not merely *observed* by an indexer but is *cryptographically enforced* by the execution of formal logic.

2.3 Determinism as a Security Boundary

A critical property of TUSM is **strong determinism**. For any given state S_t and witness Σ, there exists at most one valid next state S_{t+1}:

$$\forall S_t, \Sigma : |\{S_{t+1} \mid \delta(S_t, \Sigma) = S_{t+1}\}| \leq 1 \tag{5}$$

This eliminates branching ambiguity and the possibility of competing state interpretations. In contrast to BOSS, where two simplifiers might process the same inscription differently due to software versioning or subjective rules, a TUSM transition is mathematically singular. Any deviation in witness structure, economic parameters, or block height results in deterministic rejection.

2.4 State Commitment and the Astrolabe Pattern

To ensure **state continuity**, the new state S_{t+1} must be cryptographically committed within the destination of the spending transaction. PRECOP employs the **Astrolabe Pattern**, which binds the state directly to the Taproot output key via key tweaking (BIP-341):

$$Q_{next} = P + H(S_{t+1} \parallel \text{CMR}) \cdot G \tag{6}$$

where P is the internal public key, H is a cryptographic hash function, and G is the secp256k1 generator. The destination address Q_{next} is no longer a passive wallet address; it is the **cryptographic commitment to the future state**. Any attempt to alter the state transition (e.g., modifying the debt amount D_{t+1} or phase ϕ_{t+1}) necessarily changes the tweak and therefore the output script, causing a mismatch with the transaction's actual outputs and triggering a topology violation. This mechanism ensures that state is not merely *written* on-chain for later interpretation; it is *fused* with the spending condition itself.

2.5 Stateless Forensic Discovery: The Sovereign Beacon

The removal of external indexers introduces a new problem: how does a sovereign actor rediscover their state across the fragmented UTXO graph? PRECOP solves this via the **Protocol-Specific Probe Path**.

Instead of scanning generic BIP-84 or BIP-86 paths, the system anchors the protocol "ossements" (skeletons) at a dedicated, isolated derivation path:

$$\text{Path}_{\text{Probe}} = m/\text{precop}'/0'/0'/0/0 \dots 6 \tag{7}$$

This path acts as a lighthouse. Even if the local wallet database is lost, the actor can reconstruct the entire state history by scanning these specific "Forensic Beacons." Because the path is isolated from generic funds, the discovery process avoids the "bruit de fond" (background noise) of standard transactions, ensuring a clinical recovery of the protocol's truth.

2.6 Semantic Coherence Through State-Address Fusion

A potential criticism of the TUSM model is that without full introspection of the transaction graph (i.e., the ability to inspect other inputs and outputs directly in Script), the state machine might accept a transition that is locally valid but globally inconsistent (e.g., a debt repayment that does not actually reduce the debt because the orchestrator supplied a malformed change output). The Astrolabe Pattern resolves this by making the state commitment *holistic*. The output address Q_{next} is a function of the entire future state S_{t+1} and the program's identity CMR. If an orchestrator attempts to lie—for instance, by presenting a transaction that claims to repay a debt but in fact diverts funds—they must produce a signature that spends from an output with the correct Q_{next}. Because they cannot forge the tweak without breaking the discrete logarithm problem, any deviation from the canonical state results in an output that does not match the expected Q_{next}. The signer's local derivation of Q_{next} will fail to match the PSBT's output script, and the signing process aborts. Thus, semantic coherence is maintained not by introspecting the entire transaction, but by ensuring that any lie would require the production of a cryptographic commitment that the liar cannot compute.

2.7 TUSM vs. Indexer-Based State Models

The contrast between TUSM and the state management of meta-protocols like BOSS is stark and fundamental:

Property	**Indexer-Based (BOSS)**	**TUSM (PRECOP)**
State Location	Off-chain, virtual *MetaState*	On-chain, cryptographically committed
Validation Model	Interpretative (simplifier consensus)	Deterministic (BitMachine execution)
Consistency	Observer-dependent (subjective)	Canonical (mathematically unique)
Replay Protection	Weak, dependent on indexer logic	Strong, enforced by block height h_t
Enforcement	Social/external (trust in indexers)	Cryptographic (Taproot tweak + CMR)

By embedding the state machine directly into the UTXO graph, TUSM achieves what indexer-based systems cannot: **sovereign execution** without reliance on external validators. The **sovereign actor**'s wallet does not need to query a simplifier to know the state of their position; the state is verifiable directly from the blockchain history using deterministic, client-side logic.

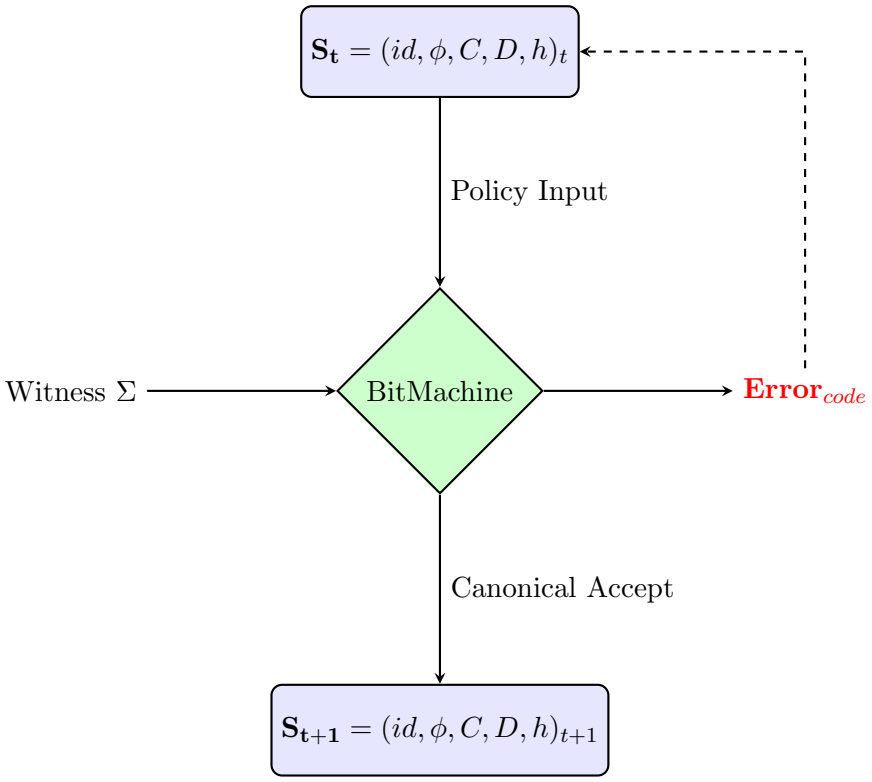

Figure 1: TUSM Transition Law: The BitMachine enforces that S_{t+1} is the unique, mathematically necessary successor to S_t.

3 Economic Invariants and Enforcement Hierarchy

3.1 Formal Mathematical Invariants

To ensure the economic safety of the **enforcement layer**, PRECOP enforces a strict set of global invariants. The most critical is the **Fee Ceiling Inequation**, which prevents unintentional value leakage during complex **covenant** execution:

$$\Delta_{Value} = \sum_{i=0}^{n} \text{In}_i.\text{value} - \sum_{j=0}^{m} \text{Out}_j.\text{value} \leq \text{MAX\_FEE\_SATS} \tag{8}$$

The constant MAX_FEE_SATS is a policy-defined upper bound on the total miner fee that may be consumed by the transaction. Any transaction whose computed fee exceeds this bound is rejected by the BitMachine before a signature is produced. This invariant is **physically inviolable**: the BitMachine evaluates the sum of input and output values using 128-bit arithmetic, and the comparison is performed as an atomic, side-effect-free jet.

Furthermore, the integrity of collateralized positions is defined by the 128-bit math safety constraints:

$$C_t \cdot P \geq D_t \cdot 150\% \tag{9}$$

where P is the oracle-provided price. Both multiplications are performed using saturation arithmetic; any overflow triggers a deterministic `ArithmeticOverflow` error, halting execution. This guarantees that the economic invariants of a CDP cannot be subverted by numeric manipulation.

The combination of the fee ceiling and the collateralization invariant ensures that the **sovereign actor** retains full control over the economic outcome of every state transition. No hidden fee, no rounding error, and no arithmetic exploit can alter the intended value flow.

3.2 Proving Zero-Panic Determinism

A fundamental requirement for institutional sovereignty is the absence of undefined behavior. We define the set of all possible execution outcomes $\mathbb{E}$ and prove that the system is fail-closed:

$$\forall \Sigma, \text{BitMachine}(AST, \Sigma) \in \{\text{Accept}, \text{Error}_{code}\} \tag{10}$$

This ensures that the **enforcement layer** never enters a non-deterministic state or a panic loop. Every rejection is accompanied by a deterministic code mapping to a ‘Topology‘, ‘Security‘, or ‘Policy‘ violation.

3.3 Enforcement Hierarchy: Tier 1 vs. Tier 2

PRECOP’s security model distinguishes two complementary enforcement layers:

Tier 1 (Consensus Enforcement): The ultimate backstop provided by Bitcoin’s proof-of-work and Taproot script validation. A transaction that violates the consensus rules (e.g., invalid signature, incorrect sighash) is rejected by the network and never included in a block. PRECOP guarantees that its derived transactions are always consensus-valid.

Tier 2 (Policy Enforcement): The set of economic and semantic rules enforced by the BitMachine *before* a signature is generated. This includes the fee ceiling, collateral ratio, output topology, and state transition invariants. Tier 2 enforcement is what prevents a signer from

ever approving a transaction that, while consensus-valid, would be economically harmful or state-inconsistent.

The current alpha release operates in what we term **Secure Shell Mode**: all Tier 2 invariants are strictly enforced locally by the BitMachine, but the ultimate commitment to these rules on-chain relies on the correct implementation of Simplicity jets (Bitcoin-specific CMRs). Until the upstream `simplicity-lang` crate provides stable, audited CMRs for all required jets, the system remains in a "fail-closed" posture—transactions are derived and validated internally, but mainnet signatures are deferred. This mode acknowledges the dependency on upstream maturation while preserving the architectural integrity of the design. The `CMR_UPSTREAM_STATUS.md` document tracks the exact status of each required jet.

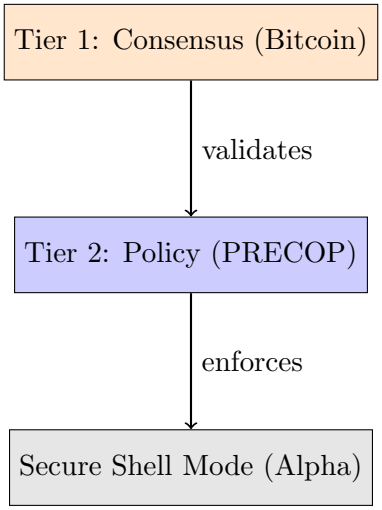

Figure 2: PRECOP Enforcement Hierarchy: Tier 2 policies are enforced locally, deferring mainnet signatures until upstream CMRs stabilize.

4 The Astrolab Linker: Deterministic Dependency Resolution

PRECOP's **enforcement layer** achieves "Zero-Variable" execution through the **Astrolab Linker**. When a **sovereign actor** proposes a transaction, the SDK resolves the underlying Simfony dependencies (e.g., 'math::safe$_m ath_1$28')$into a flattened, linear AST$.

4.1 The Astrolab Purity Seal: Environment-Free Dependency Resolution

A critical security property of the **Astrolab Linker** is its immunity to environmental side effects. PRECOP's linker enforces the **Astrolab Purity Seal**:

> *All module resolutions are performed against a content-addressed, sealed registry. The inclusion of a jet such as `math::safe_math_128` is not resolved via a filesystem path but via its immutable cryptographic hash (CMR).*

This guarantee ensures that a contract authored today will resolve to the exact same bytecode and CMR a decade from now, regardless of the host operating system, compiler version, or available network resources.

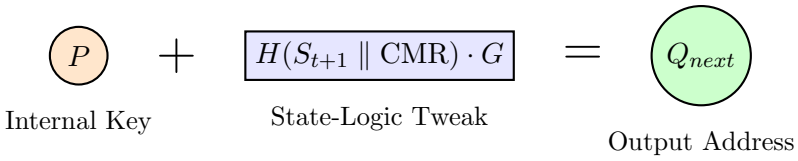

Figure 3: State-Address Fusion: The Bitcoin address Q_{next} is no longer a destination, but a cryptographic proof of the future state S_{t+1} and program identity CMR.

5 Canonical Output Topology: The Command-First Pattern

To eliminate the "Blind Signature" vulnerability, PRECOP enforces a strict, index-based output sequence as the **Law of the Protocol**. The structure of the transaction is not merely a container; it is the message itself.

5.1 The Command-First Ordering Rule

Every PRECOP-governed transaction must adhere to the following strict topology, optimized for fail-fast evaluation and semantic clarity:

1. **Output 0: Metadata (OP_RETURN).** A zero-value envelope containing the protocol discriminant and witness commitments. Placing this output first allows the Astrolab Linker to immediately validate the contract identity before expending computational resources on value calculations. Any mismatch triggers an instant `TopologyViolation`.

2. **Output 1: Primary Target (Vault/Recipient).** The continuation of the state machine or the primary objective of the debt/payment cycle.

3. **Output 2: Relay/Change (Owner/Seller).** The return of collateral or the change from a trade, anchored to the sovereign actor's identity.

4. **Output 3: Treasury Fee (Protocol Siphon).** A policy-bounded fee redirect, typically used for protocol maintenance or treasury funding.

Any deviation from this sequence—whether by injection, reordering, or omission—triggers a `TopologyViolation` in the BitMachine, and no signature is produced. This renders index-shift attacks and "fee siphoning" mathematically impossible. The command-first design also reinforces the **Header-Payload Invariant**: the transaction begins with its governing law.

Command-First Topology: OP_RETURN at index 0 enables immediate policy validation.

Figure 4: Canonical Output Topology (Command-First): A fixed sequence that eliminates indexer ambiguity and signer blindness.

6 Matrix of Sovereign Use Cases

Use Case	TUSM Mechanics	Key Invariant
Sovereign CDP	Phase MINT → REPAY	$C \cdot P \geq D \cdot 150\%$ (128-bit)
Atomic Swap L1	Hash-lock + Timelock	'jet::eq$_2$56' + '$jet :: check_locktime$'
MEW (Sovereign Wallet)	Governance Branch (3/5)	Isolation MAST (Leaf 0)
Universal BRC-20	State-Address Fusion	Q_{next} commit of Total Supply

6.1 Case Study: Atomic Swap on Bitcoin L1

The **Atomic Swap** is a canonical example of a trustless cross-chain or cross-asset exchange. Traditional implementations rely on HTLCs (Hashed Timelock Contracts) that require cooperation between two parties and often involve off-chain coordination. PRECOP enables a fully sovereign, L1-native atomic swap where the enforcement of the swap logic is embedded directly in the UTXO state machine.

6.1.1 TUSM State Representation

An atomic swap instance is defined by the following TUSM state:

$$\begin{aligned} S_t = (&id_t, \\ &\phi_t \in \{\text{OPEN}, \text{CLAIMED}, \text{REFUNDED}\}, \\ &\text{amount\_A}, \text{amount\_B}, \\ &\text{hashlock}, \text{timelock}, \\ &h_t) \end{aligned}$$

6.1.2 Simplicity Enforcement

The covenant logic is expressed in Simplicity using two critical jets:

- `jet::eq_256`: Verifies that the witness provides a preimage x such that SHA256(x) = hashlock.
- `jet::check_lock_time`: Enforces that the transaction's `nLockTime` is greater than or equal to the specified `timelock`.

The spending conditions are encoded as two distinct MAST leaves:

1. **Claim Path (Leaf 0)**: Requires the witness to contain a valid preimage x. Upon successful verification, the state transitions to ϕ = CLAIMED, and the output commits to the new state where the claiming party receives the swapped asset.
2. **Refund Path (Leaf 1)**: Can only be executed after the `timelock` has expired. It does not require the preimage and transitions the state to ϕ = REFUNDED, returning the funds to the original owner.

6.1.3 Canonical Derivation in Practice

When a **sovereign actor** wishes to claim the swap, their PRECOP-enabled wallet does not simply sign a PSBT provided by a counterparty. Instead, it:

1. Reconstructs the expected transaction topology from the policy.
2. Executes the Simplicity AST with the provided preimage witness.
3. Verifies that `jet::eq_256` returns `true` and that the state transition is valid.
4. Computes the canonical sighash and compares it with the PSBT.

Only if all steps succeed is a signature produced. The **sovereign actor** is mathematically guaranteed that their funds will only move if the correct preimage is revealed, and that they cannot be cheated by a malicious PSBT that attempts to bypass the hashlock or timelock constraints.

This L1-native atomic swap eliminates the need for external indexers or trusted relayers. The entire logic is enforced by the BitMachine, with state continuity guaranteed by the Astrolabe Pattern.

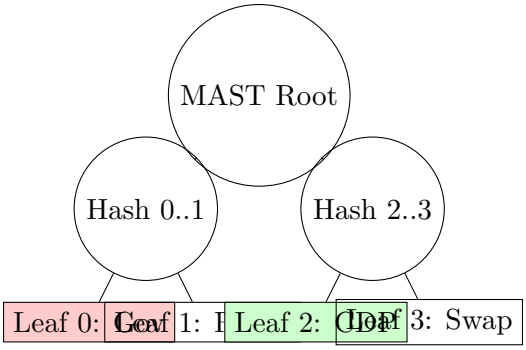

Figure 5: Selective Disclosure via MAST: Only the CDP branch is revealed, preserving the privacy of other sovereign intents.

7 Conclusion: From Observed State to Enforced Sovereignty

The PRECOP framework establishes a new paradigm for programmable finance on Bitcoin. By shifting the security boundary from *post-hoc verification* to *pre-signature canonical derivation*, PRECOP eliminates the PSBT Trust Problem and the indexer-dependency inherent in meta-protocols such as BOSS. The system's three foundational pillars—the Template-Enforced UTXO State Machine (TUSM), the Astrolabe Pattern for state-address fusion, and the Canonical Engine with BitMachine execution—operate in concert to ensure a single, mathematically provable invariant:

> *There exists exactly one valid transaction for a given policy and state. Any deviation results in deterministic rejection.*

7.1 Summary of Contributions

This work makes the following concrete contributions to the field of Bitcoin covenant research:

1. **Formalization of TUSM:** A rigorous mathematical model for embedding state machines directly within Bitcoin's UTXO graph, complete with replay protection, domain isolation, and economic consistency proofs.
2. **The Astrolabe Pattern:** A Taproot-based commitment scheme that cryptographically fuses the next state to the destination address, eliminating the possibility of output redirection attacks.
3. **Canonical Derivation Protocol:** A deterministic algorithm for reconstructing transactions from first principles, rendering the signer immune to PSBT manipulation and fee siphoning.
4. **Fail-Closed Execution Environment:** Integration of the Simplicity BitMachine with a strict error taxonomy, ensuring that undefined behavior is impossible and that all failures are auditable.
5. **Consensus Equivalence Proof:** Differential fuzzing methodology and byte-for-byte parity with `rust-bitcoin`, guaranteeing that PRECOP introduces no consensus divergence.

7.2 The End of Indexer Subjectivity

The contrast with indexer-driven frameworks like BOSS is not merely technical but philosophical. BOSS offers a *descriptive* layer: it observes inscriptions and attempts to construct a consistent narrative. PRECOP offers a *prescriptive* layer: it enforces that only the correct narrative can be written to the chain. In a BOSS-based system, a **sovereign actor** must trust that their simplifier has correctly interpreted the latest JSON payload. In a PRECOP-governed system, the **sovereign actor's** own device mathematically proves the correctness of the state transition before the transaction is even broadcast. This is the essence of **Programmable Sovereignty**: the **sovereign actor's** keys control the funds, and the **sovereign actor's** local execution controls the logic.

7.3 Future Work and Upstream Dependencies

The current PRECOP v0.1.0-alpha release is feature-complete with respect to its core derivation and validation logic, as documented in the `RELEASE_MANIFEST.md`. However, production readiness on Bitcoin mainnet is contingent upon the maturation of the `simplicity-lang` Rust crate. Specifically:

- **Stable Bitcoin Jet CMR Support:** The current upstream version (0.7.0) lacks stable CMR derivation for Bitcoin-specific jets. PRECOP enforces a **Canonical CMR Strictness** policy; until upstream support is available, the system operates in a **Secure Shell Mode** where execution is proven but mainnet signatures are deferred. The `CMR_UPSTREAM_STATUS.md` file tracks this dependency.
- **Jet Weight Calibration:** Accurate cost modeling for Simplicity jets is required for fee estimation. This is expected in the 0.8.x release series.

Upon resolution of these upstream items, PRECOP will be positioned for a formal third-party security audit, as outlined in the `AUDIT_CLOSURE_PLAN_V70.md`.

7.4 Final Remarks

Bitcoin was conceived as a system for peer-to-peer electronic cash, but its potential as a substrate for sovereign financial contracts has long been constrained by the limitations of passive scripting. PRECOP demonstrates that with deterministic derivation, formal semantics, and a fail-closed architecture, it is possible to achieve **covenant enforcement without consensus changes**. We do not seek to alter Bitcoin; we seek to align our tools so perfectly with its existing rules that invalid states become not merely unconfirmed, but *impossible to sign*.

Vires in Numeris.

Acknowledgments

The authors acknowledge the foundational work of the Simplicity team at Blockstream, the `rust-bitcoin` maintainers, and the broader Bitcoin research community exploring the frontiers of covenants and stateful computation.

Audit and Provenance

This document corresponds to PRECOP release v0.1.0-alpha (Git commit `6ed4f9a`). All referenced audit artifacts—including the Threat Model, Invariants, and Evidence Package—are available in the project repository. The release has been internally verified against the 10-Point Audit Closure Plan.

References

[1] Pieter Wuille, Jonas Nick, Anthony Towns. *BIP-341: Taproot: SegWit version 1 spending rules*. Bitcoin Improvement Proposal, 2020.

[2] Pieter Wuille, Jonas Nick, Anthony Towns. *BIP-342: Validation of Taproot Scripts*. Bitcoin Improvement Proposal, 2020.

[3] Russell O'Connor. *Simplicity: A New Language for Blockchains*. Blockstream Whitepaper, 2017.

[4] Andrew Chow. *BIP-174: Partially Signed Bitcoin Transaction Format*. Bitcoin Improvement Proposal, 2018.

[5] The Ord. *BOSS Lightpaper: Bitcoin Operational Standard System*. 2023.

[6] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. 2013. URL: `https://ethereum.org/en/whitepaper/`.

[7] Laz1m0v. *Audit Closure Plan, Evidence Package, and Invariants (v0.1.0-alpha)*. Internal Documentation, 2026.