Zable

A Verifiable Asset Overlay

ZABLE PROTOCOL
Architecture, verification model, and Genesis policy

Abstract

Zable proposes an asset protocol whose state is derived by applying explicit, versioned rules to ordered records carried by Zcash transactions. The design separates ownership authorization, encrypted note discovery, asset policy, and settlement observation. Its central objective is reproducible interpretation: parties with the same authenticated history and rule version should be able to derive and compare the same asset state without accepting an indexer's conclusions on trust.

This paper defines the system model, explains Zable addresses, and distinguishes permissionless participation from issuer authority and application-level selection. It also specifies the current policy for an 8,192-item Genesis PFP collection: equal allocations to a fixed-price sale and a weighted whitelist draw, with fee-distribution and governance-participation utilities. Independent replay is conditional on authentic inputs and complete rules; it is not, by itself, proof of settlement finality, unbiased selection, or operator performance.

Keywords: asset overlays; Zcash; non-fungible tokens; deterministic replay; encrypted ownership; weighted allocation

1. Introduction

A digital asset system must make two separate questions answerable: who may authorize a transition, and why should another observer accept the resulting state? Zable addresses these questions through cryptographic ownership and a deterministic state transition model. The carrier chain supplies transaction ordering and settlement evidence; the asset protocol supplies interpretation. No additional validator consensus is assumed.

The intended benefit is inspectability. An asset identifier, a transfer, a sale, and a payment condition should have explicit meanings that can be checked against an ordered history. "Permissionless" therefore describes access to protocol participation, not freedom to override another issuer's policy or bypass an application's allocation rules.

This document is a protocol-design proposal. It defines intended semantics and the evidence needed to validate them; publication of the paper does not itself establish a production deployment, a security audit, or a completed independent verification.

2. Context and related standards

ERC-721 standardizes interfaces for NFT ownership and transfer in an Ethereum execution environment. Zable uses a different execution model: ownership transitions are interpreted from carrier records rather than Ethereum contract calls. The comparison is functional, not a claim of ABI or network compatibility. [1]

ERC-2981 standardizes how a royalty recipient and amount can be queried; it does not require every transfer or marketplace to pay royalties. This distinction also matters for Zable: a royalty rule can constrain a defined settlement path without controlling all transfers, side payments, or external agreements. [2]

Zcash memo fields provide the carrier-data context. ZIP 302 describes a 512-byte memo field and interpretation conventions, including binary data. A Zable envelope uses that capacity to carry protocol records; it does not enlarge the carrier's consensus rules or grant an overlay the ability to reverse a valid carrier payment. [3]

3. System model

The model contains five roles. A *carrier* supplies ordered transactions and settlement data. An *issuer* defines an asset and its issuance policy. A *wallet* holds authorization secrets and discovers owned notes. An *indexer* reconstructs state. An *application operator* coordinates activities such as Genesis selection, refunds, or holder distributions. One party may perform multiple roles, but their authorities remain distinct.

$$S_h = \delta_v(S_{h-1}, B_h) \tag{1}$$

Here S_h is the asset state after height h, B_h is the complete relevant ordered input for that height, and δ_v is the transition function for rule version v. Determinism requires the same initial state, ordering, decoding rules, arithmetic, and authenticated inputs. An HTTP response bearing a block identifier is not automatically an authenticated block.

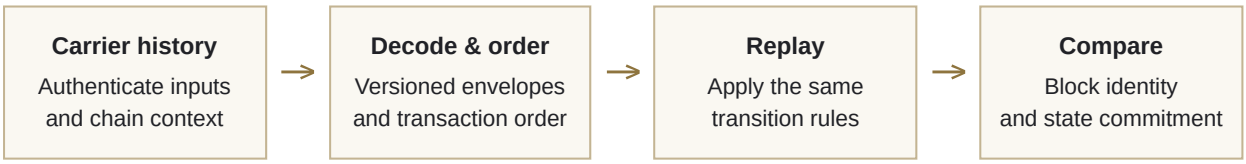


Figure 1. Re-execution checks interpretation only after the input-authentication boundary has been satisfied.

The model assumes secure cryptographic primitives, correctly implemented key handling, and access to the data needed for validation. Carrier finality follows the carrier's own security assumptions. Off-chain promises require separate evidence and accountability; they are not converted into consensus guarantees by recording a commitment.

4. Why a Zable address?

A Zable address identifies the public keys used by the asset layer. It carries a spending public key A and an encryption public key E. The first supports authorization; the second lets a sender encrypt the information a recipient needs to discover and later spend a note. A public address is shareable. Its seed and secret keys are not.

$$A = aB, \quad E = dB, \quad \text{address} = \text{Encode}(A \,\|\, E) \tag{2}$$

B denotes the group generator, while a and d are separately derived secret scalars. Zable uses ristretto255 group encodings and a Bech32m-style address encoding with its own human-readable prefix. These choices provide a representation and checksum; they do not make a Zable address a Bitcoin or Zcash payment destination. [4] [5]

4.1 Local derivation and compatibility

A compatible client derives both key pairs from a securely generated 32-byte seed using the same domain-separated Zable derivation rules. Re-entering that seed under those rules reproduces the same base address. Browser and command-line clients must share the derivation and encoding specification; generic wallet-seed compatibility is not implied.

The registration interface is designed to perform seed-to-address derivation in browser memory. Seeds must not be submitted to a form, sent to a server, or written to local storage, session storage, or cookies. A user-controlled backup is necessary because the service cannot recover a missing seed. Memory-only handling reduces persistent exposure but does not protect against a compromised device, browser extension, or page.

4.2 Note ownership

For a recipient's spending key A, a sender can construct a one-time note key $K = A + tB$, using a fresh scalar t. The corresponding spending secret is $k = a + t$. An ownership commitment binds K to fresh note material ρ; spending discloses the required opening and a signature over the authorized operation.

Encrypted note data conveys discovery material to the recipient. Authenticated encryption protects confidentiality and integrity subject to key and nonce discipline; this is a cryptographic component, not a guarantee of transaction anonymity. The design uses ChaCha20-Poly1305 for authenticated encryption and domain-separated BLAKE2 hashing. [6] [7]

> **Address submission is not proof of ownership.** A form associates an X identity with a public Zable address, but possession of the corresponding secret requires a separate cryptographic challenge or authorized operation. The address must never be treated as a ZEC deposit address.

Disclosure is contextual: the sender knows the recipient and the construction material, while public asset identifiers, quantities, and transition graphs may remain observable. From initial registration, the form provider and operator can associate an X identity, public Zable address, and submitted post URL.

5. Protocol records and asset state

5.1 Canonical envelopes

A protocol payload is serialized canonically and divided into memo-sized chunks. The envelope contains a binary marker, protocol magic, version, payload identifier, chunk index, chunk count, and declared data length. With a 44-byte header, a 512-byte memo accommodates up to 468 bytes of chunk data. Decoders must reject unsupported versions and malformed encodings consistently.

Completed records are ordered by their carrier position: height, transaction index, and output index, using the location at which the final required chunk becomes available. Boundary actions such as expirations must also have an explicit order. A signature commits to the canonical operation representation with signature fields excluded as specified by the signing rules.

5.2 Identifiers and authority

An asset identifier binds an issuer public key and an asset name under a domain-separated hash. This separates issuer namespaces: another issuer cannot acquire the identical identifier by choosing the same name. It does not prevent a misleading name, copied artwork, or impersonation at the presentation layer.

Transition family	Required authority and meaning
Issue / Mint	An issuer creates an asset policy and authorizes supply under its limits. Open asset creation is not authority to mint another issuer’s asset.
Transfer / Burn	Owners authorize consumption of notes. Transfers preserve permitted asset quantities or NFT identifiers; burns remove the authorized units.
Offer / Cancel / Accept	Offers lock specified assets and define exchange conditions. Cancellation and acceptance depend on current offer state and ordering.
MintSale / MintClaim	An issuer establishes a sale policy. Claims must meet its supply, payment, timing, and any configured eligibility conditions.
Bid / Refund / CloseSale	Auction records express bids, settlement bookkeeping, and closure according to sale rules; refund performance has an operator boundary.
Migrate	Reserved for a future design. It supplies no present migration transition or bridge guarantee.

5.3 Invalid operations and conservation

An invalid operation must not produce its intended asset or sale transition. “Inert” does not mean that nothing happened anywhere: replay tracking and diagnostic records may remain, and a carrier payment can remain valid even when the overlay operation fails.

Conservation checks cover the quantities or NFT sets consumed and produced by each authorized transition. Supply increases must follow issuer authority, including authority delegated through an issuer-created sale. Validation must also examine interactions among multiple operations in the same carrier transaction; correct isolated checks alone do not establish complete economic correctness.

6. Settlement, royalties, and distributions

6.1 Payment-conditioned asset transitions

A ZEC-denominated acceptance or mint claim can require qualifying payment outputs in the same carrier transaction. For an operation to succeed, observed amounts must satisfy the recipients and amounts required by that operation. Obligations to the same recipient must be aggregated correctly, and transaction-wide accounting must prevent incompatible reuse of settlement evidence.

The implication is one-directional: a successful asset transition requires its payment conditions. The reverse implication does not follow. If an offer is already consumed, a sale is exhausted, or another ordered condition fails, the carrier may still process the payment. An overlay cannot undo that payment merely by rejecting the asset operation.

$$\text{Valid asset settlement} \Rightarrow \text{required payment evidence}$$
$$\text{Payment evidence} \not\Rightarrow \text{guaranteed asset delivery} \tag{3}$$

Consequently, a payment-conditioned operation must not be described as a universally atomic, bidirectional ZEC–asset swap. Applications need explicit handling for stale state, concurrent claims, and failed delivery. Exchanges whose two asset legs are both controlled by the same reducer have a different boundary: both asset-state changes can be accepted or rejected together.

6.2 Royalties are path-specific

A defined ZEC sale path can require a royalty computed from the sale amount and an asset's basis-point policy. That requirement applies where the protocol explicitly validates it. A plain transfer, an asset-for-asset exchange, and an external agreement are not automatically equivalent to that sale path.

Royalty metadata and royalty enforcement must therefore be described separately. A policy can make a designated settlement operation invalid without its royalty output, while still being unable to observe or compel every economic agreement involving the asset. This is consistent with the distinction between royalty information and payment execution in ERC-2981. [2]

6.3 Operator obligations

In a prepaid auction model, funds sent to an operator-controlled payment address create custody exposure. A deterministic refund ledger can make amounts due inspectable, and a refund record can validate a qualifying payment after it occurs. Neither mechanism forces an operator to execute a refund.

Similarly, a sale's fixed-recipient fee splits are not a general holder-distribution system. Genesis fee distribution requires a defined revenue pool, rate, holder-eligibility or snapshot policy, calculation method, and payment process. Under an operator-executed model, publication of the calculation improves accountability but does not make payment automatic or trustless.

Governance participation is a separate proposed utility. Its decision scope, voting method, eligibility, delegation, and execution authority require a published governance policy. Ownership alone does not define those mechanisms.

7. Verification and reproducibility

7.1 Authenticate first, replay second

An independent verifier must establish which carrier history is relevant, authenticate its chain context, obtain the complete required records and payment evidence, and apply the selected rule version. Replaying server-supplied JSON can check internal consistency, but cannot alone establish that the JSON describes canonical Zcash transactions or correctly decoded outputs.

A useful checkpoint identifies at least the network, rule version, height, block identity, and state commitment. Comparing only a height and digest can conceal a disagreement about chain history. A commitment must cover every field that can affect future transitions; excluding such a field weakens the meaning of a matching digest.

7.2 What reproducibility establishes

Property	Evidence required
Deterministic replay	Identical initial state, complete inputs, rule version, and canonical interpretation produce identical checkpoints.
Independent execution	Another party obtains the necessary data and runs validation independently. Reusing the same library does not constitute an independent implementation.
Conformance	Reviewed fixtures include expected state and rejection behavior. Missing reference outputs must fail validation, not be silently regenerated.
Reproducible builds	Pinned toolchains and dependencies, a defined build environment, and repeated artifact comparisons support byte-level build claims.

These properties are complementary, not interchangeable. Passing a fixture does not authenticate a live chain. Rebuilding source does not prove byte-identical releases. Agreement among copies of one implementation does not exclude a shared defect. Strong assurance requires release provenance, independent scrutiny, and test coverage appropriate to the claimed behavior.

7.3 Content commitments and availability

Content-addressed blobs and Merkle roots can bind metadata or artwork to particular bytes. They answer an integrity question: whether supplied content matches a commitment. They do not ensure that all referenced content is published, retrievable, or retained. A collection must specify which content is committed, when the commitment is fixed, and how the bytes remain available.

A commitment also does not establish fair artwork assignment or a reproducible generative pipeline. Those claims need their own inputs, algorithms, parameters, and publication procedure. Concept artwork should not be confused with a finalized collection-generation specification.

7.4 Reorganizations and recovery

When the carrier history changes, an indexer must return to a verified common state and replay the replacement history. If a required checkpoint lies outside retained snapshots, recovery requires an earlier validated checkpoint or a complete replay. Key recovery, history recovery, and data availability are separate responsibilities.

8. ZABLE Genesis

Genesis is planned as a single collection of **8,192 PFP NFTs** with equal per-NFT rights. It has no separate privileged tier. Its proposed utilities are fee distribution and participation in governance decisions; holding Genesis is not an exclusive license to participate in future protocol minting.

4,096	4,096
FIXED-PRICE SALE · 0.4096 ZEC EACH	WL WEIGHTED-DRAW ALLOCATION

Equal per-NFT fee units are the proposed distribution basis. The revenue definition, payment policy, and governance rules require separate specification; neither utility implies guaranteed income or an already defined voting mechanism.

8.1 Registration and draw application are separate

Stage 1 — WL registration. The initial Google Form requires an X handle, public Zable address, one URL to the applicant's own published post tagging **@zablenfts**, and follow/post confirmations. These submissions record identity and evidence; they do not automatically enter the NFT draw or establish a verified point.

Stage 2 — NFT draw application. A separate form opens later for the same identity and address plus multiple post URLs. Include the initial post if eligible; submitting it in both forms earns no extra points. The operator calculates credited points from evidence and manually checks selected candidates' follow status and posts.

8.2 Weighted participation

Each eligible post tagging @zablenfts earns one point, with at most one credited post per day. Accumulated eligible-post points determine draw weight. Multiple posts or duplicate submissions do not increase the daily limit.

$$w_i = \Sigma_d \min(1, e_{i,d}) \tag{4}$$

In (4), $e_{i,d}$ counts eligible posts for participant i on day d under posting-period and day-boundary rules to be announced. The equation defines weight, not a sampling algorithm. An overall allocation probability cannot be inferred from a single-draw weight ratio without the full selection method.

8.3 Policy publication and accountability

The fixed-price allocation costs 0.4096 ZEC per NFT; this does not establish a price for the WL allocation. The eligible posting period, daily timezone, application dates, sale and mint timing, WL mint cost, multi-account rules, maximum wins, cross-pool participation, and handling of unfilled or unclaimed allocations remain to be specified. No fixed WL deadline or sale date is established here.

A publicly reproducible weighted draw requires a published eligible-input record, scoring rules, sampling algorithm, randomness procedure, and verification artifacts, with appropriate personal-data safeguards. The social weighted draw is distinct from bid ranking and any optional sale-assignment rules. An organizer-signed eligibility voucher can prove organizer authorization; it cannot prove that the underlying social review was correct or unbiased.

9. Security and trust model

Permissionless asset creation means that participation need not depend on a central issuer allowlist. It does not remove issuer control of supply, sale-specific gates, operator selection, or the need to protect private keys. Each claim should identify both the rule being checked and the authority it does not eliminate.

The principal trust boundaries are input authenticity, wallet execution, cryptographic implementation, data availability, and operator performance. A viewing capability must be evaluated for its actual disclosure scope; a diversified payment address does not by itself imply that a viewing key reveals exactly one payment. Public records and off-chain identity submissions can remain linkable.

Reproducibility makes specified computations inspectable. It does not prove economic incentives, prevent every implementation defect, compel refunds or distributions, or turn subjective eligibility decisions into objective facts. Formal proofs, security audits, and independent implementations are distinct forms of evidence and must be identified by their actual scope.

10. Roadmap

The post-mint direction is sequential: **launchpad → NFT marketplace → collaboration with other NFT projects to support migration to Zable**. A launchpad extends issuance workflows; a marketplace extends discovery and exchange; partner migration requires project-specific asset, metadata, and holder-entitlement arrangements.

Migration assistance is not a claim of an existing trustless bridge. A future migration design must specify source-chain authentication, custody or burn/lock assumptions, destination issuance authority, and conflict handling before any such guarantee can be evaluated.

References

1. Ethereum Improvement Proposals. *ERC-721: Non-Fungible Token Standard.* https://eips.ethereum.org/EIPS/eip-721
2. Ethereum Improvement Proposals. *ERC-2981: NFT Royalty Standard.* https://eips.ethereum.org/EIPS/eip-2981
3. Zcash Improvement Proposals. *ZIP 302: Standard Memo Field Format.* https://zips.z.cash/zip-0302
4. Internet Research Task Force. *RFC 9496: The ristretto255 and decaf448 Groups.* December 2023. https://www.rfc-editor.org/rfc/rfc9496
5. Bitcoin Improvement Proposals. *BIP 350: Bech32m format for v1+ witness addresses.* https://github.com/bitcoin/bips/blob/master/bip-0350.mediawiki
6. Internet Research Task Force. *RFC 8439: ChaCha20 and Poly1305 for IETF Protocols.* June 2018. https://www.rfc-editor.org/rfc/rfc8439
7. Internet Engineering Task Force. *RFC 7693: The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC).* November 2015. https://www.rfc-editor.org/rfc/rfc7693