Move value. Move data. On your terms.

A ZooBC transaction is a signed envelope wrapping a small typed body. The **envelope** is identical for every type — the **body** and a few optional **modifiers** decide what actually happens: a plain send, a token transfer, a vested salary, an escrowed deal, an app move, a scheduled trigger. This is the full map.

01 Anatomy of a transaction

Everything is serialized little-endian into one byte buffer, then signed. The blue fields are the **envelope** (same for all types); the purple **body** is what changes per type.

`type` 4 | `version` 1 | `timestamp` 8 | `sender` 4+key | `recipient` 4+key / empty | `fee` 8 | `bodyLen` 4 | `body` bodyLen | `escrow` empty / section | `msgLen` 4 | `message` msgLen | `signature` on send

An **account address** is `4-byte type (LE) + public key` — `00000000`+32B ed25519 for ZBC, `04000000`+ETH, `05000000`+BTC, and so on. A missing recipient is written as the **empty** type `02000000`. The **message** tail is where a plain send carries its memo — the body of a `SendZBC` is *only* the amount; the recipient and memo live in the envelope.

02 Signing

The account type decides both the address prefix and the signature scheme. Sign over the whole envelope (without the signature field). All three are deterministic, so a browser wallet and the node agree byte-for-byte.

ZooBC (ed25519) · type `00000000`

```
// digest, then sign
digest = SHA3_256(txBytes)
sig    = ed25519(digest, privkey)   // 64 bytes
```

Also the scheme for Solana, Cardano, Tezos, Polkadot — same ed25519 key, different address prefix + display encoding.

Ethereum (secp256k1) · type `04000000`

```
// recoverable ECDSA, low-s
digest = keccak256(SHA3_256(txBytes))
r,s,v  = secp256k1.sign(digest)
sig    = r ‖ s ‖ (v+27)          // 65 bytes
```

MetaMask can also send its own signed RLP; the node recovers the sender and maps it to a ZBC transfer.

Bitcoin (secp256k1) · type `05000000`

```
// double-sha over the SHA3 payload
digest = SHA256d(SHA3_256(txBytes))
sig    = secp256k1.sign(digest)
field  = [len(2)] ‖ compressedPub(33) ‖ sig
```

Fee, in short

The minimum fee is size-aware: a base rate × the network **fee-scale**, plus a per-byte storage/duration component. Read it live from `/api/v1/fee-params` or price an exact tx with `/estimate-fee` so the wallet never disagrees with the node to the atomic unit. Overpayment on a precise-fee send is refunded.

03 The three kinds of coin

Amounts are always **atomic** (1 ZBC = 100,000,000). The only thing that changes between coins is the `token_id` field — the movement mechanics are identical.

① Native ZBC

`token_id = 0` everywhere. The gas/fee coin, block rewards, staking. A `SendZBC` (type 1) is the plain path.

② Genesis / bridge tokens

Wrapped foreign assets minted at genesis (ZBTC, ZETH, ZSOL…). They are ordinary **colored tokens** with a fixed `token_id` assigned in the genesis config — nothing special at the protocol level.

③ User-issued tokens

Anyone mints one with `IssueToken` (type 10): symbol, supply, decimals, optional ZBC backing. Its `token_id` derives from the issue tx. Same `TransferToken` / `MintToken` / `BurnToken` apply.

Wherever a body has a `token_id` (transfer, swap, app stake, liquid, vesting…), pass `0` for ZBC or the token's id for a colored coin. Backed tokens can even pay their own fee (a `fee_in_token` flag), priced against their ZBC backing.

04 Modifiers — one payment, many shapes

Beyond the base types, three timing/trust wrappers change *when* and *whether* funds actually land. They are what make the surface feel large.

Escrow

Append an **escrow section** to *any* value tx's envelope (approver · commission · timeout · instruction). The funds are held until the named **approver** signs `ApprovalEscrow` (type 4) to release or reject — or the timeout expires. A conditional deal on top of any transfer.

Liquid

`LiquidPayment` (type 6): a single **delayed** payment over a window. The recipient is paid the full amount when the window completes, or a pro-rata slice if the sender `LiquidPaymentStop`s (type 262) early. Short-term (hours), ZBC or token.

Scheduled / Vested

`ScheduledTransfer` (type 29): release in **tranches on a timer** — vesting (funds pre-locked) or recurring salary (pulled each period). Cliff, revoke, recipient-decline, reassign. Long-term (weeks–years). The releases show up as ledger **movements**, not new txs.

Trigger / Stop

The lightweight timer: `CreateTrigger` (15) locks an amount to fire at a future height; `CancelTrigger` (16) refunds it. To stop the long ones: `CancelSchedule` (30) revoke/decline, `LiquidPaymentStop` (262).

They stack. A monthly salary *that is also escrowed*, a vesting grant in a colored token — the primitives compose. (Full composition of recurring-with-escrow is on the roadmap; the building blocks are all here today.)

05 Keeping data-objects alive

On-chain objects that hold data — **datasets** (key/value profiles, chat, business forms) and **tokens** — pay rent to persist. "Adding funds to an object" means topping up its survival financing.

Datasets & files → prepaid storage

You write data with `SetupAccountDataset` (type 3, a `property → value` pair) or store a file with `DFSCreateFile` (8). Each block bills storage rent against your account's **prepaid storage**; when it runs dry the object is pruned. Top it up with `AddPrepaidStorage` (type 9) — body is just an amount, credited straight to your prepaid-storage balance. More prepaid = longer life.

Tokens → persistence financing

A colored token also pays to survive. `FinanceToken` (type 14) tops up a token's persistence horizon from the fee — extending how long it stays live before its unused backing is returned to the creator. Same idea, per-token.

06 Full transaction catalog

All 50 transaction types, with their exact little-endian body layout. coin = supports token_id · modifiable = accepts an escrow section · new chain = consensus-level (ships with a chain launch).

SendZBC

1 · modifiable · live now

Plain transfer. Recipient + memo ride in the envelope.

`amount` 8

LiquidPayment

6 · coin · modifiable · live now

Delayed payment over a window; pays at completion (or pro-rata on stop).

`amount` 8 `complete_minutes` 8 `token_id` 8

LiquidPaymentStop

262 live now

Stop an active liquid payment; settle pro-rata now.

`liquid_tx_id 8`

ScheduledTransfer

29 coin new chain

Vesting or recurring payment released in tranches on a timer.

`token_id 8` `per_fire 8`
`interval_s 8` `remaining_fires 4`
`cliff_s 8` `funding_mode 1`
`cancel_policy 1` `end_time 8`
`reserved 1`

CancelSchedule

30 new chain

Sender revoke (if revocable) or recipient decline; unfired funds → funder.

`schedule_id 8`

ReassignSchedule

31 new chain

Recipient redirects a schedule's future tranches to a new address.

`schedule_id 8` `new_recipient addr`

CreateTrigger

15 live now

Lock an amount to be released at a future block height (or oracle event).

`fire_height 8` `amount 8`
`event_id? lp4`

CancelTrigger

16 live now

Cancel a pending trigger; refund the locked amount to the owner.

`trigger_id 8`

AttestEvent

17 live now

Oracle attestation: a node posts an external event’s value on-chain.

`event_id` lp4 `value` lp4

IssueToken

10 live now

Create a colored token: symbol, supply, decimals, optional ZBC backing.

`decimals` 1 `flags` 1 `total_supply` 8 `backing` 8 `symbol` lp2 `name` lp2

TransferToken

11 coin modifiable live now

Move a colored token between accounts.

`token_id` 8 `amount` 8 `fee_in_token?` 1

MintToken

12 coin live now

Mint more of a mintable token (issuer only).

`token_id` 8 `amount` 8 `fee_in_token?` 1

BurnToken

13 coin live now

Burn (destroy) token units; redeemable tokens release backing.

`token_id` 8 `amount` 8 `fee_in_token?` 1

FinanceToken

14 coin live now

Top up a token’s persistence financing so it survives longer.

`token_id` 8

SetupAccountDataset

3 live now

Write a property→value on your account (profile, chat, business forms).

`property` lp4 `value` lp4

RemoveAccountDataset

259 live now

Remove a dataset entry.

`property` lp4 `value` lp4

AddPrepaidStorage

9 live now

Top up your prepaid-storage balance so data-objects keep paying rent.

`amount` 8

DFSCreateFile

8 live now

Store a file in the decentralized file store.

`path` lp4 `content_len` 4 `content` n

DFSUpdateFile

264 live now

Replace a stored file’s contents.

`path` lp4 `content_len` 4 `content` n

DFSDeleteFile

520 live now

Delete a stored file.

`path` lp4

EscrowRequest

260 live now

Recipient-initiated escrow: propose a held transfer needing approval.

`proposed_sender` addr `amount` 8 `approver` addr `commission` 8 `timeout` 8 `instruction` lp4 `expiry?` 8

ApprovalEscrow

4 live now

Approver approves / rejects / expires a held escrow.

`decision` 4 `escrow_tx_id` 8

MultiSignature

5 live now

Create a multisig account, or carry + co-sign an inner transaction.

`present` 4

`min · nonce · participants` var

`inner_tx` lp4 `signature_info` var

CreateSwapOffer

18 coin live now

Offer give-token for want-token (give held in escrow). P2P swap.

`give_token` 8 `give_amt` 8

`want_token` 8 `want_amt` 8 `expiry` 8

`counterparty?` 36

AcceptSwapOffer

19 live now

Fill an open swap offer (atomic swap + fees).

`offer_id` 8

CancelSwapOffer

20 live now

Cancel your open swap offer; refund the held amount.

`offer_id` 8

CreateMarket

21 live now

Open a permissionless (base,quote) order-book market.

`base_token` 8 `quote_token` 8

`rent_deposit` 8

PlaceOrder

22 live now

Limit/market buy or sell on a market’s book.

`market_id` 8 `side` 1 `price` 8

`amount` 8 `flags` 1 `expiry` 8

CancelOrder

23 live now

Cancel a resting order; refund the held remainder.

`order_id` 8

CreateApp

24 coin live now

Open an app — type, stake, seats, params, optional opponent (direct challenge).

`app_type` 1 `stake_token` 8 `stake_amount` 8 `seats` 1 `params` lp2 `opponent?` 36

JoinApp

25 live now

Join an open app seat, locking an equal stake.

`app_id` 8

AppMove

26 live now

Make one move; rules enforced on-chain.

`app_id` 8 `move` lp2

ResignApp

27 live now

Forfeit; your stake share goes to the opponent(s).

`app_id` 8

ClaimAppTimeout

28 live now

Claim the win when an opponent abandons past the deadline.

`app_id` 8

NodeRegistration

2 live now

Register a validator node (locked stake + proof of ownership).

`node_pubkey` 32 `owner` addr `locked_balance` 8 `proof_of_ownership` 72+64

NodeRegistrationUpdate

258 live now

Update a node’s locked balance / registration.

`node_pubkey` 32 `new_locked` 8 `proof_of_ownership` n

RemoveNodeRegistration

514 live now

Remove a node from the validator set.

`node_pubkey` 32

ClaimNodeRegistration

770 live now

Claim a previously-removed node registration.

`node_pubkey` 32 `proof_of_ownership` n

FeeVoteCommitment

7 live now

Commit phase of the network fee-scale vote (hash).

`vote_hash` 32

FeeVoteReveal

263 live now

Reveal phase of the fee-scale vote.

`block_hash` 32 `block_height` 4 `fee_vote` 8 `voter_sig` lp4

StoreFile

40 new chain

Anchor a decentralized-storage file: its manifest (root + piece ids) + a rent deposit. The content lives off-chain, sharded + replicated; only the manifest is on-chain.

`file_root` 32 `total_size` 8
`piece_size` 4 `deposit` 8
`piece_count` 4 `piece_ids` 32×n

SubmitStorageProof

41 new chain

A replica attests possession of a challenged piece this epoch; tallied on-chain for the rent reward / slash. Emitted by the storage node, not a user.

`piece_id` 32 `nonce` 8 `proof` 32

SettleApp

39 new chain

Settle a state-channel app on-chain: replay the ordered signed move vouchers through the rules and pay the winner. move_count is capped at 1024 vouchers and must equal final_seq; a count larger than the body can hold is rejected before any allocation.

`app_id` 8 `final_seq` 4 `move_count` 4
`vouchers[]` seat1+move.lp2+sig64

RegisterGateway

36 new chain

Register a gateway (key + domain + url) in the on-chain registry, locking a stake.

`gateway_key` 32 `domain` lp4 `url` lp4

GatewayHeartbeat

37 new chain

Liveness proof — an unforgeable, tip-fresh reference-block hash. Emitted by the running gateway process, not user-built.

`ref_block_hash` 32

UnregisterGateway

38 new chain

Withdraw a gateway from the registry; refund the locked stake.

`gateway_key` 32

RegisterRelease

32 new chain

Publish a signed binary release (version + manifest hash), signed by the release authority.

`version` lp4 `manifest_hash` 32 `release_address` addr

RevokeRelease

35 new chain

Revoke a previously-published release (e.g. a bad build).

`version` lp4

ReleaseAuthorityPropose

33 new chain

Current authority proposes handing the release role to a new address.

`new_authority` addr

ReleaseAuthorityAccept

34 new chain

The pending address accepts, becoming the new release authority.

`(empty body)`

Byte layouts are transcribed from the ZooBC node serializers (`transaction_util.cpp`, the per-type executors, and the archival decoder). `lp4` = 4-byte length prefix + UTF-8 bytes; `addr` = 4-byte type + key; all integers little-endian; amounts atomic (÷1e8 for ZBC). Reference for integrators building wallets and tooling on ZooBC.