POD 0005: The Idiomatic Odin API Surface

COMMITTED

CATEGORY	**STATUS**
Design Record	Committed
AUTHORS	**CREATED**
Vikrant Rathore <vikrant@insan.ai>, Paxos Odin Contributors	2026-09-16
LAST UPDATED	**LABELS**
2026-09-17	api, odin, naming, design

Abstract

This record captures the decisions that shaped the public surface of `paxos-odin`: how types and error values are spelled, how a call site chooses between a proc-group verb and a receiver-prefixed procedure, how per-node tuning is passed, how `Effects` is declared and sized, and which alternatives were tried and rejected. The `0.1.0` decisions are kept as written; the section "The 0.2.0 decisions" adds what the data-oriented redesign (POD 0009) and rotating ownership (POD 0010) changed, and the before-and-after tables cover both steps. The goal throughout was a surface that reads as ordinary Odin, keeps every capacity visible in a type, and lets the compiler catch mismatches that used to be runtime asserts.

Status and Implementation Boundary

The API decisions defined in this document are implemented; the versioned before-and-after tables remain historical design records. Membership canonicalizes ids in ascending order, so callers need not agree on input order. Binary search uses that same array; there is no separate membership index array.

Recovery scratch is chunk-sized and candidate selection is frozen before entering phase two; these remain internal layout optimizations. Effects buffers continue to borrow payloads until the next transition. Ownership admission probes target slots before mutating state, and `resubmits_dropped` exposes any overflow in its best-effort queue. POD 0011 defines a separate Python API over a C boundary without altering these Odin conventions.

Naming

Types and error values

Types are `Ada_Case` with underscores between words: `Node_Id`, `Trim_Anchor`, `Promise_Range_Message`, `Replicated_Log_Node`, `Durability_Gate`. Error values in `Error` are spelled the same way: `.Not_Leader`, `.Window_Full`, `.Configuration_Mismatch`, `.Log_Sealed`. The pre-release code used `NodeId` and CamelCase error values (`.InvalidNodeId`); the rename made the enum consistent with the message and write type names it appears next to. `0.1.0` kept `NodeId`, `Log_Slot`, and `Vote_Ballot` as compatibility aliases; `0.2.0` removed them, since `Node_Id` changed width and `Ballot` changed representation (below), and an alias would have hidden both.

Procedures

Every procedure has a receiver-prefixed long spelling: `node_propose`, `replicated_log_propose_stop_sign`, `learner_learn_chosen`, `effects_messages_slice`, `membership_init`, `ledger_apply`. The prefix is the type the first argument points to, so a reader of a call site knows the receiver without looking up the type. On top of those, `src/paxos.odin` declares one proc group per verb:

```
init    :: proc{node_init, effects_init, membership_init,
                replicated_log_init, learner_init, stop_sign_init}
propose :: proc{node_propose, replicated_log_propose}
step    :: proc{node_step, replicated_log_step, replicated_log_step_checked}
tick    :: proc{node_tick, replicated_log_tick}
ledger  :: proc{node_ledger, replicated_log_ledger}
```

A host therefore writes `paxos.init(&node, id, membership)` and `paxos.init(&log, id, configuration_id, membership)` with the same verb; the argument types pick the procedure. The `log_*` aliases (`log_propose`, `log_reconfigure`, `log_step`) exist for hosts that only use the replicated log and want short names that still say which receiver they take. The `Effects` accessors have bare short spellings (`reset`, `confirm_writes_durable`, `writes_slice`, `messages_slice`, `committed_slice`, `requests_slice`, `requires_power_loss_barrier`, `pre_durable_messages`, `is_empty`) because there is only one receiver for them. The `Ledger` procedures (`ledger_apply`, `ledger_replay_fold`, `ledger_vote_at`, `ledger_chosen_at`, …) keep their long spelling only: a host meets them at replay, where naming the receiver is the point.

Node_Options

The pre-release surface had `node_init_with_priority`, `node_restore_with_priority`, `node_restore_at`, and log counterparts such as `replicated_log_restore_with_priority`. Each added one positional parameter and multiplied the number of entry points. They were replaced by one struct, which `0.2.0` extended by one field:

```
Node_Options :: struct {
  priority:                          u8,
  election_timeout_ticks:            u32,
  heartbeat_interval_ticks:          u32,
  resend_interval_ticks:             u32,
  gate_proposals_on_inherited_prefix: bool,
  campaign_disabled:                 bool,
  rotating_ownership:                bool,
}
```

Zero means the default for every field, so `node_init` takes `options := Node_Options{}` and `or_default` substitutes `DEFAULT_ELECTION_TIMEOUT_TICKS`, `DEFAULT_HEARTBEAT_INTERVAL_TICKS`, and `DEFAULT_RESEND_INTERVAL_TICKS` for zero tick counts. `node_restore`, `node_continue_at`, `replicated_log_init`, `replicated_log_restore`, `replicated_log_continue_at`, and `replicated_log_init_from_stop` all take the same trailing `options`. The lifecycle procedures also agree on argument order: the node id comes first, the configuration id (for the log) second, and a `floor` precedes an `anchor` wherever both appear. `priority` is a `u8` because it occupies eight bits of the packed ballot; `rotating_ownership` selects the mode of POD 0010 at initialisation, and `sim/simulation.odin` and `bench/main.odin` pass it straight from their command line.

Effects

- **Same parameters as Node.** `Effects(Value, MAX_MEMBERS, WINDOW_SLOTS, CHUNK_SLOTS, GATE)` mirrors `Node`; every transition names the two with identical parameter lists, so an `Effects` sized for a different chunk fails to compile (`effects_must_match_node` in `tools/check_contracts.py`).

- **Tight capacities.** The four `small_array` lists are sized to the per-transition maxima given in POD 0002, and since `0.2.0` an `Envelope` and a `Write` have a fixed size whatever the payload (POD 0009), so the batch no longer grows with `size_of(Value)`.
- **Zero is initialised.** The gate flag is `writes_pending`, set by `effects_add_write` and cleared by `effects_confirm_writes_durable`, so a freshly declared `Effects` needs no call before its first transition and cannot be accidentally "unlocked" by re-initialising it. `effects_init` remains as the documented unchecked reset for a batch the process will never complete.
- **Comparable values.** `Node`, `Effects`, `Ledger`, and `Learner` carry `where intrinsics.type_is_comparable(Value)`, and equality is native `==`. Odin equality compares the active union variant and string contents, unlike the raw memory comparison used before the first review.
- **Remembered no-op.** `Node.noop` is `Maybe(Value)`; it is set by `campaign` and `tick` and consulted by `maybe_resolve_chunk`, which returns `.Missing_Noop` rather than filling a hole with a zero value.
- **Log_Envelope.** `Log_Envelope(Value, MAX_MEMBERS, MAX_METADATA_BYTES)` pairs a `configuration_id` with a core `Envelope(Entry(...))`. `replicated_log_envelope` stamps one and `replicated_log_step_checked` (reachable through the `step` group) checks it. POD 0006 gives the protocol reasons.

Design Decisions in 0.2.0

`Node_Id` is `u16`

A ballot needs the proposer's id to be unique, and the redesign packs the ballot into one integer. Sixteen bits leave forty for the round and eight for the priority, and `MAX_SUPPORTED_MEMBERS = 65535` is larger than the `128` that `0.1.0` allowed. The `0.1.0` alternative "larger membership bound", previously rejected when acknowledgements relied on a 128-bit native `bit_set`, has now been adopted: `acknowledgements` is `[WINDOW_SLOTS]Bit_Set(MAX_MEMBERS)`, the array-backed set from `src/bit_set.odin`, and `review_thousand_voters_reach_quorum` exercises a membership above `LINEAR_LOOKUP_LIMIT`.

`Ballot` is one packed `u64`

`Ballot :: distinct u64` with `ballot_make(round, priority, node)` and the accessors `ballot_round`, `ballot_priority`, `ballot_node` replaces the three-field struct and `ballot_less_than`. Call sites compare ballots with `<`, `max`, and `==` directly, which is what Lamport's B1 asks for. The `distinct` keyword prevents a `Slot` or raw `u64` from being inadvertently passed as a ballot. `BALLOT_ZERO` is the empty promise, and round zero is reserved for slot owners (POD 0010).

`Ledger` replaces `Durable_State`

The durable state a host restores is `Ledger(Value, WINDOW_SLOTS)`: columns rather than an array of `Durable_Cell`. Its procedures are the `ledger_` group (`ledger_apply`, `ledger_replay_fold`, `ledger_cell`, `ledger_vote_at`, `ledger_chosen_at`, `ledger_is_chosen`, `ledger_claim`, `ledger_highest_ballot`, `ledger_highest_used`), and the node exposes it through `node_ledger` / `replicated_log_ledger` and the `ledger` proc group. `restore` and `restore_learner` take a `Ledger` by value. POD 0009 explains the layout.

Records and messages carry `^Value`

`Write_Vote(V){ballot, slot, value: ^V}`, `Write_Chosen(V){slot, value: ^V}`, `Committed(V){slot, value: ^V}`, and the three value-bearing messages point into the producing node's ledger. `Write_Promise_At{ballot, slot}` is new (a per-decree promise), `Write_Trim` is the anchor, and `Trim_Anchor` lost its `history_hash` (the host binds the image checksum to `trim_id`). `message_value(message)` returns the pointer and whether the kind carries one, so a transport needs no `switch` of its own to copy the payload.

The `Packet` idiom

An in-process transport that queues envelopes across transitions declares

```
Packet :: struct($Value: typeid) {
  envelope: paxos.Envelope(Value),
  value:    Value,
}
```

with `packet_of(envelope)` copying the payload at enqueue and `packet_envelope(&packet)` repointing the message at the packet's copy for the `step` call. The idiom is written out in `tests/harness.odin`, `examples/counter.odin`, `sim/simulation.odin`, and `bench/main.odin` rather than shipped in `src/`: the library has no opinion about queues, and a host with a real codec never needs it. The journal-side twin is `Journal_Record{write, value}` with `journal_append` and `journal_replay` in the test harness.

`Prepare_Scope` and `Nack_Message.slot`

`Prepare_Message` gained `scope: Prepare_Scope` (`.Global`, `.Bounded`) so that a revocation can promise a range of decrees without disturbing the global promise, and `Nack_Message` gained `slot` so an owner can tell which decree was refused. Both are zero-default, so a `.Global` prepare and a prepare nack are spelled as before.

Before and After

Pre-release to 0.1.0

Pre-release	0.1.0
`NodeId`	`Node_Id` (`NodeId` kept as an alias)
`.InvalidNodeId`, `.NotLeader`	`.Invalid_Node_Id`, `.Not_Leader`
`node_init_with_priority(&n, id, m, 2)`	`paxos.init(&n, id, m, paxos.Node_Options{priority = 2})`
`node_restore_at(...)`, `node_restore_with_priority(...)`	`paxos.restore(&n, id, m, durable, floor, options)`
`Host_Managed_Node(Value, ...)` wrapper struct	`Node(Value, M, W, C, .Host_Managed)`
`Effects` with its own capacity parameters	`Effects(Value, M, W, C, GATE)` mirroring `Node`
`effects_init` required before first use	zero value ready; `effects_init` for abandoned batches
`node_step(&n, e, &fx)` only	`paxos.step(&n, e, &fx)` and the long spelling

0.1.0 to 0.2.0

0.1.0	0.2.0
`Node_Id :: u32`; `NodeId`, `Log_Slot`, `Vote_Ballot` aliases	`Node_Id :: u16`; aliases removed
`Ballot :: struct{round: u64, priority: u32, node}`, `ballot_less_than`	`Ballot :: distinct u64`, `ballot_make`, `<` and `max`
`Node_Options.priority: u32`	`Node_Options.priority: u8`, plus `rotating_ownership`

`Durable_State(Value, W)` with `cells: [W]Durable_Cell(Value)`	`Ledger(Value, W)` with columns and bitmaps
`durable_apply`, `durable_replay_fold`	`ledger_apply`, `ledger_replay_fold`
`replicated_log_durable_state`	`replicated_log_ledger`; `ledger` proc group
`Write_Promise`, `Write_Accept`, `Write_Commit`, `Write_Trim_Anchor`	`Write_Promise`, `Write_Promise_At`, `Write_Vote`, `Write_Chosen`, `Write_Trim`
`Accept_Message{ballot, slot, value: Value}`	`Accept_Message{ballot, slot, value: ^Value}`
`Promise_Message{ballot, slot, accepted: Accepted(Value)}`	`Promise_Message{ballot, slot, vote, state, value: ^Value}`
`Committed{slot, value: Value}`	`Committed{slot, value: ^Value}`
`Trim_Anchor{trim_id, chosen_trim_slot, history_hash}`	`Trim_Anchor{trim_id, chosen_trim_slot}`
`Prepare_Message{ballot, first}`	`Prepare_Message{ballot, first, last, scope}`
`Nack_Message{rejected, promised, decided_through}`	`Nack_Message{rejected, promised, slot, decided_through}`
any positive `WINDOW_SLOTS`	`WINDOW_SLOTS` a power of two (`#assert`)
`MAX_SUPPORTED_MEMBERS = 128`	`MAX_SUPPORTED_MEMBERS = 65535`
envelopes queued as values	`Packet{envelope, value}` copied at enqueue

Alternatives Considered

- **A wrapper `Log_Effects` type.** A separate effects struct for `Replicated_Log_Node` would have hidden the `Entry` union behind a friendlier name, but it would have needed its own accessors and its own gate, and the host would have had two batch types with identical semantics. The log instead documents the declaration `Effects(Entry(Value, MAX_MEMBERS, MAX_METADATA_BYTES), MAX_MEMBERS, WINDOW_SLOTS, CHUNK_SLOTS, GATE)` on `Replicated_Log_Node`, and the tests alias it once (`Seal_Effects` in `tests/test_reconfiguration_sim.odin`). POD 0009 records why the redesign rejected it a second time.
- **Untyped option literals through proc groups.** `paxos.init(&n, id, m, {priority = 1})` does not compile. Overload resolution for a proc group needs the type of every argument, and an untyped compound literal has none until a target type is known; the group cannot choose `node_init` in order to learn that the fourth parameter is `Node_Options`. The call must spell the type: `paxos.init(&n, id, m, paxos.Node_Options{priority = 1})`. The long spelling `node_init(&n, id, m, {priority = 1})` accepts the untyped literal because it is a single procedure. This is the price of the proc-group surface and is documented rather than worked around.
- **A `Host_Managed_Node` wrapper with `using`.** Removed. The gate is a type parameter, so a plain `Node(..., .Host_Managed)` states the exception at the declaration, and the four host rules live on `Durability_Gate` where a reader of the enum sees them.
- **Larger membership bound (0.1.0).** The native `bit_set` for acknowledgements capped `MAX_MEMBERS` at 128, and a wider bound was judged not worth the indirection. `0.2.0` reversed this: the packed ballot fixed the id width at 16 bits, and the array-backed `Bit_Set` was already the window bitmap, so acknowledgements use it too.
- **A `Packet` type in `src/` (0.2.0).** Shipping the in-process copy idiom would have put a queue policy into a library that owns no transport. The four host programs each spell the twelve lines; a host with a codec never needs them.

- **Keeping `NodeId` and the ballot struct as aliases (0.2.0).** An alias `NodeId :: Node_Id` would have compiled old code whose ids no longer fit in sixteen bits, and there is no alias that turns a three-field struct into a `distinct u64`. Both were dropped so the compiler reports every site that needs attention.

References

- POD 0002: Paxos-Odin: Architecture and Pure State Machine Design.
- POD 0003: Durability Contracts, Window Reuse, and Trim Anchors.
- POD 0006: Reconfiguration and Epoch Isolation.
- POD 0007: Review Findings and Verification Evidence, sections "API changes" and "Third pass".
- POD 0009: The Data-Oriented Ledger.
- POD 0010: Rotating Slot Ownership.
- Odin language overview: procedure groups, parametric polymorphism, `distinct`, `Maybe`, and `bit_set`.