ASSET MANAGER — WHAT IT DOES

It locks, schedules, and releases GENc.
No *maybe*. No *almost*. Either the timestamp is due and tokens move, or it isn't and they don't.

ROLES & BOUNDS

- **Owner (deployer):** the only address allowed to fund pools.
- **GENERAL (bot):** the only address allowed to start the clock and execute releases.
- **Token:** immutable GENc interface bound at deployment.
- **Reentrancy:** blocked. Transfers use `SafeERC20`.

ORIGINS & CAPS (HARD-WIRED)

Each pool has a fixed recipient and a hard cap. Funding can never exceed the cap — any attempt reverts.

ID	Pool	Address	CAP
1	Team	`0x318cFAe23438428501086EDe36f07DAc0f8Ffec4`	5 000 000 000 GENc
2	Marketing	`0x2801dd36E26c376a110Baa8C16C19411719117cf`	5 000 000 000 GENc
3	Dev	`0x251611603459649D89b724342c08E9d5c93E55C5`	1 000 000 000 GENc
4	Expo/Event	`0x64b55A3612f9BEDD447c1C8BeaDE6BB7D708eFf3`	3 000 000 000 GENc
5	Future Project	`0xB148f668a560f8193Ff566DbB8471626fE2B6818`	3 100 000 000 GENc
6	SBA Reserve	`0x15BaF29c43d9Cbf1dDE1300f2A268322F6635E77`	3 100 000 000 GENc

THE CLOCK

- **WasteStart (once):** set by GENERAL after Whitelist Close → `WasteStartSet(ts)`.
- **Time unit:** `DAY = 60 s` (test mode). On mainnet, 1 day = 24 hours.
- The schedule is **gap-based**, not block-based — time is absolute, not relative.

FUNDING (OWNER-ONLY)

Owner provides tokens to each pool.
Functions: `FUND_TEAMGENc`, `FUND_MARKETINGGENc`, `FUND_DEVGENc`, `FUND_EVENTEXPO`, `FUND_FUTUREPROJECT`, `FUND_SBA`.
Transfers pull from the owner wallet and respect the hard cap.
Emits `PoolFunded(id, amount, newFunded, origin)`.

RELEASE (GENERAL-ONLY)

`release(id)` executes one scheduled step if due.

- If the timestamp has passed → tokens transfer, index advances, event emits.
- If not yet due or completed → nothing happens.
  Emits `ReleaseExecuted(id, index, amount, origin, dueTs)`.

SCHEDULE — HARD FACTS ONLY

Time base (test build): `DAY = 60` → 1 minute = 1 day.

All gaps count forward from `wasteStart`. Nothing moves before it. Nothing moves if it's not due.

ID 1 — Team

- **Amounts:** 500 M TGE, then 300 M × 15 → 16 releases, total 5 B.
- **Gaps:** `tgeGap = 0`, `cliffGap = 100`, `stepGap = 30`.
- **Timeline:** Day 0 (10 %), Day 100 (+6 %), then every 30 days × 14 (each +6 %) → complete Day 520.
- **Share:** 10 % + (15 × 6 %) = 100 %.

ID 2 — Marketing

- **Amounts:** 500 M TGE, then 300 M × 15 → 16 releases, total 5 B.
- **Gaps:** `tgeGap = 0`, `cliffGap = 75`, `stepGap = 30`.
- **Timeline:** Day 0 (10 %), Day 75 (+6 %), then every 30 days × 14 (each +6 %) → complete Day 495.
- **Share:** 10 % + (15 × 6 %) = 100 %.

ID 3 — Dev

- **Amounts:** 200 M TGE, then 100 M × 8 → 9 releases, total 1 B.
- **Gaps:** `tgeGap = 90`, `cliffGap = 50`, `stepGap = 50`.
- **Timeline:** Day 90 (20 %), Day 140 (+10 %), then every 50 days × 7 (each +10 %) → complete Day 490.
- **Share:** 20 % + (8 × 10 %) = 100 %.

ID 4 — Expo / Event (corrected)

- **Amounts:** 1 B × 3 → three equal yearly releases; **no TGE at Day 0**.
- **Gaps:** `tgeGap = 365`, `cliffGap = 365`, `stepGap = 365`.
- **Timeline:**
  - Day 365 (+ 1 B, ≈ 33.33 %)
  - Day 730 (+ 1 B, ≈ 33.33 %)
  - Day 1095 (+ 1 B, ≈ 33.34 %)
- **Share:** ≈ 33.33 % + 33.33 % + 33.34 % = 100 %.

ID 5 — Future Project

- **Amounts:** Single unlock 3.1 B → 1 release.
- **Gaps:** `tgeGap = 365` (no cliff, no steps).
- **Timeline:** Day 365 → 100 % at once.
- **Share:** 100 %.

ID 6 — SBA Reserve

- **Amounts:** 600 M TGE, then 500 M × 5 → 6 releases, total 3.1 B.
- **Gaps:** `tgeGap = 0`, `cliffGap = 50`, `stepGap = 50`.
- **Timeline:** Day 0 (≈ 19.35 %), Day 50 (+ 16.13 %), then every 50 days × 4 (each + 16.13 %) → complete Day 250.
- **Share:** ≈ 19.35 % + (5 × 16.13 %) = 100 %.

STATE & READOUTS — TRANSPARENCY BY DESIGN

- `nextDue(id)` → next timestamp, index, amount.
- `scheduleLen(id)` / `scheduleAt(id,i)` → slice amount + time + done flag.
- `adviseRelease(id)` → checks if due.
- `poolStatus(id)` → cap, funded, released, sealed, length, next due, amount.
- `currentDaySinceStart()` → monotonic counter since WasteStart.
- All pools track funded, released, nextIndex, lastReleaseTs.

GUARANTEES

- Immutable routing — recipients and caps are hard-coded.
- Split authority — Owner funds, GENERAL releases.
- Deterministic timing — pure gap math from recorded timestamps.
- No minting, no overflow — moves only what was funded.
- Full traceability — every action emits an event; the ledger is the audit trail.

WHAT THIS ENFORCES

Recipients and caps cannot be changed.
No early access. No bypass.
No manual unlocks. No shortcuts.
Owner can fund but never release. GENERAL can release but never fund.
Gap math runs on its own clock — **time, not trust, decides.**
No backroom switches.
No human adjustments mid-stream.
You see the pools, the schedule, the index, the next due, and the transaction when it hits.
If someone wants to argue with that, they’ll be arguing with the timestamps.
The chain doesn’t care.
The chain doesn’t blink.

— ProofBeforePromise