Corp-OS Architecture

A folder of markdown files, and a set of skills that operate on it under rules the folder itself declares. Everything below is either **a place data sits**, **a door it moves through**, or **a rule about what may cross which door** — and the rules are the product. Without them this is a notebook that nobody trusts by March.

The shape of it

Marketecture — what sits where

Read top to bottom: material enters from the world, lands in a layer that is never edited, passes a gate a person controls, becomes knowledge that can be scanned, and leaves through a boundary that logs what it removed. The rail on the right is the control plane — the files and scripts that govern all of it and belong to none of it.

THE WORLD — Where things get said

outside the OS

`Meetings & calls` `Chat threads` `Email` `Documents` `Issue trackers` `Research & the web` `Your own notes`

None of this is knowledge yet. It is **things people said**, in systems that will not remember why they mattered.

↓ THREE DOORS, AND ONLY THREE

INGRESS — Getting material in

writes to raw/ autonomously

corp-os-pull
Fetches what is new from sources you registered, deduped on `external_id`.

corp-os-intake
Files something you hand over — a paste, a transcript, a doc, a note.

corp-os-migrate
Brings an existing body of work across once, at founding, with original dates.

corp-os-connect registers a source before any of this: its protocol, its cadence, what it feeds, what medium it produces — and what it is blind to. That last field is why a gap in coverage is visible rather than assumed away.

↓ WRITTEN, NEVER REWRITTEN

CONTROL PLANE

Read first by every skill, on every run. None of it is content.

config.json
The authority: which layers exist, what they are called, their roles and schemas, decay windows, retention policy, gate strictness. Skills fall back to shipped defaults only where it is silent.

meta.json
Counts, ID cutoffs, and a dated history of what was done to this OS.

usage/log.md
One line of friction per run. This is the evidence `corp-os-improve` reads — the OS improves from what happened, not from opinion.

scripts/
Nine scripts your OS carries its own copies of, so it keeps working when the plugin is not loaded.

STORAGE — LAYER 1

`raw/` — the append-only source archive

role: source

One file per thing said
`YYYY-MM-DD--<source>--<slug>.md`, content essentially as given: placed and tagged, not rewritten.

Frontmatter carries the provenance
source, person, date, type, jobs, tags, `external_id`, `processed`.

Never edited, never deleted
One sanctioned exception: flipping `processed: true`. Deletion only under a declared retention obligation, and it leaves a tombstone.

Existing here means *said*, not *true*. That distinction is the whole reason the layer is separate. Everything downstream can be thrown away and rebuilt from this; this cannot be rebuilt from anything.

↓ NOTHING CROSSES WITHOUT A PERSON

THE GATE

`proposals/` — written to disk before it is shown to you

role: record

A file, not a chat message
`PROPOSAL-YYYY-MM-DD-<slug>.md`: the headline, then each addition with its provenance and a recommendation.

Outcomes appended after
confirmed · declined · deferred · modified. The declines are the valuable part.

Promoting skills
`claims` `jobs` `decide` `glossary` `company`

The gate is not there because any one entry is risky. It is there because **without it the derived layer stops being knowledge and becomes a second, messier copy of `raw/`**. A proposal that lives only in a session dies with it, and leaves no record of what you chose not to know.

↓ CONFIRMED, AND ONLY THEN

THE CORRECTION LOOP

`corp-os-reality-check` works the **decay window** on every entry: past its date, never verified, contradicted, or an assumption still worn as a fact.

Default 90 days, scoped to entries with a retrievable source, `none` for durable facts, 30 for metrics. Without it a knowledge base ages into confident fiction, because each entry still looks reasonable on its own.

THE WORKBENCH

Acting on the OS itself rather than on what it holds. Not a sequence.

`guide` `setup` `migrate` `configure` `upgrade` `rebuild` `improve` `audit` `pattern` `contribute`

ACROSS PEOPLE

A **pattern** is a portable spec for producing one kind of output — a dashboard, an export, a deck. It addresses layers **by role and field, never by name**, so it survives an OS that renamed its vocabulary.

Binding produces three outcomes and no fourth: bound, bound with drops named out loud, or refused with the missing role stated. An empty panel is never an outcome — it reads as a state rather than a defect.

STORAGE — LAYER 2

The derived layer

role: derived · fully regenerable

`claims/`
The unit of knowledge. One statement, its kind, confidence, fidelity, verbatim citation, decay window.

`jobs/`
The outcomes you are working toward, each with what it still needs to know.

`decisions/`
Forks still open: owner, `decide_by`, what they block, what would settle them.

`company/ · glossary.md`
Employers, counterparties, competitors; internal jargon and metric definitions.

`arguments/`
Optional. A conclusion and what it rests on, rendered as `N entries / M sources`.

anything you declare
`experiments/`, `matters/`, `readouts/` — with a schema and an index line, treated like a shipped layer.

Regenerable is the point. `corp-os-rebuild` can re-derive all of this from `raw/`, which is what makes it safe to restructure. A layer whose role is `record` — you maintain it by hand — is never touched by a rebuild.

BESIDE IT, NOT IN IT

`sensitive.md` — quarantine, outside the scan path

role: record

Sensitivity is **two axes**, and conflating them made the OS quietly wrong. `sensitivity` is the export class — what must not leave. `bearing` is whether the OS can reason correctly without it. Only **incidental** sensitive material is quarantined here; **load-bearing** sensitive material stays in the scan path, marked, and is stripped at the export boundary. Quarantining everything sensitive does not produce a gap — it produces a confidently wrong answer with nothing in it to signal the omission.

↓ COUNTED BY A SCRIPT, NOT BY A MODEL

ACCESS

INDEX.md — the scannable surface

generated by build_index.py

One line per entry
A descriptor, never a bare link — a link forces a second read, a one-liner answers most questions outright.

Per-layer indexes
Any layer over the threshold gets its own, so a layer holding most of the corpus still has an enumerated entry point.

Cross-cutting views
Open evidence across every job, oldest first. Arguments resting on one source. Entries one call from promotion.

No layer may be enabled without declaring **what one entry contains** and **what one line of it looks like here**. That rule is what keeps this file worth reading first.

↓ OUT, WITH PROVENANCE ATTACHED

EGRESS

Getting value back out

reads the scan path

corp-os-recall
Answers a question with citations, or folds what the OS knows into what you are writing.

corp-os-brief
The recurring operating brief: what moved, what needs a decision, what went stale. Schedulable.

corp-os-dashboard
Binds a pattern and runs its generator. Defaults to a local file, because the scan path holds sensitive material on purpose.

corp-os-redact
The export boundary. The cleaned copy and a private log of every removal are written together, or neither is.

Sharing is an export event, not a mode. Corp-OS assumes one operator; the moment something leaves, it crosses a boundary that records what it took out. A screen-share shield protects a screen and a redacted build protects a file — ship both, and let neither imply the other.

Using it, from a standing start

Flow — which skill, and when

Two forks decide everything at the beginning; after that it is a loop of four moments. Skill names below are what you ask for; the cyan pills are slash commands, which cover the moments that recur often enough to deserve one.

FORK 1

NOTHING EXISTS YET

corp-os-setup

An interrogation across nine areas — role and mandate, why you want this, which data earns its keep, how information reaches you, what you can connect, your company, your output needs, your sensitivity boundaries. Fifteen to twenty minutes, and it is the deliverable as much as the folder is: a scaffold built without it produces a generic notebook that gets abandoned in a month.

`/corp-os`

YOU ALREADY HAVE A PILE

corp-os-migrate

Bringing an existing body of work across has four failure modes and they all happen on day one: no cohort ceiling on confidence, original dates replaced by today's, every decay window firing on the same day, and no record of what you deliberately left behind. Decide what does *not* come across before anything is written.

↓ THEN THE LOOP

MOMENT 1

Something was said, and it will matter later

Registered source? `corp-os-pull` fetches it and dedupes. Handed to you in a paste or a file? `corp-os-intake` files it. Either way it lands in `raw/` with its frontmatter, and the skill writes there without asking — capture is the one direction that needs no gate, because nothing in `raw/` claims to be true.

`/corp-os-capture` `/corp-os-catchup` `connect — register the source first`

↓

MOMENT 2

Turn the pile into things you can cite

Work the unprocessed queue. Each candidate becomes a proposal on disk with its kind, confidence, source fidelity and verbatim citation attached, and *you* confirm, decline, defer or modify. A decision already made is a claim; a fork still open is a `decision` record with an owner and a date past which it gets made by default.

`claims` `decide` `/corp-os-open` `jobs` `glossary` `company`

↓

MOMENT 3

YOU HAVE A QUESTION

corp-os-recall

Answered from the index down, with citations. Or pointed at whatever you are drafting, to fold in the claims, terms and history that bear on it.

`/corp-os-recall`

YOU WANT A STANDING VIEW

corp-os-brief · corp-os-dashboard

A brief is prose on a cadence and can schedule itself. A dashboard is a page you return to, built by binding a pattern rather than hand-writing counts that are wrong within a day.

`/corp-os-brief`

↓

MOMENT 4

Take things out again

Monthly-ish, sweep what has drifted: past its decay window, never verified, contradicted by something newer, or an assumption that has been quietly promoted to a fact. This is the invariant most systems lack, and it is what decides whether yours survives two years — an OS answering with confidence it has not earned since March is worse than no OS, because now you believe it.

`/corp-os-check` `reality-check`

↓ AND WHEN SOMETHING LEAVES

BOUNDARY

Make it safe to send

Personal, personnel, compensation, credential and confidential material comes out, and a private log of exactly what was removed is written in the same call as the cleaned copy. Entries that share a citation are checked against each other first: per-entry redaction was mechanically correct across 836 claims of a real corpus and the boundary still leaked, withholding one copy of a quote and emitting the identical text in full twice.

`/corp-os-share` `redact`

↩ **And back to the top.** What you sent out is itself material — the deck, the brief, the summary you pasted into a thread. It gets captured like anything else.

OFF THE LOOP, ON A LONGER CYCLE

The shape stops fitting → `corp-os-configure` renames vocabulary, adds or disables layers, declares a custom one, retunes decay or the gate — treating anything that needs a migration as a migration, with enumeration and a confirmation first.

You updated the plugin → `corp-os-upgrade`, because updating the plugin does not update your OS. Your OS carries its own script copies; this refreshes them by content, stamps the version, and names the structural migrations it refuses to perform for you.

It has started to annoy you → `corp-os-improve` reads the friction lines in your usage log and proposes changes with counts attached. Structure nobody uses gets retired; the same manual step every run becomes a field.

Your team's outputs all look different → `corp-os-pattern` turns one of them into a portable spec and packs it for everyone else.

How data gets in

Three doors, one destination

All three write to the same place in the same shape. What differs is who initiated it and how much arrives at once — which is exactly what decides whether a confidence ceiling has to be set before anything is minted.

DOOR	INITIATED BY	WHAT IT WRITES	THE FAILURE IT EXISTS TO PREVENT
`corp-os-pull`	You, on a cadence	One `raw/` file per new item, deduped against `external_id`; then proposals	Re-filing the same meeting three times because two connectors saw it
`corp-os-intake`	You, ad hoc	One `raw/` file, placed and tagged rather than rewritten, plus a `placement:` instruction if you gave one	A summary landing where the source should be, with nothing left to cite
`corp-os-migrate`	You, once	The whole cohort into `raw/` with *original* dates, a cohort ceiling set before minting, then staggered decay windows	410 entries all coming due on the same day, which teaches you to skip the sweep permanently

THE RULE UNDERNEATH ALL THREE

A skill may write to `raw/`, `INDEX.md`, `meta.json` and `usage/log.md` on its own. It may only **propose** anything entering a layer whose role is `derived`. The asymmetry is deliberate: capture is cheap to undo and knowledge is not.

How data is stored

Markdown, on your disk, readable without any of this

There is no database and no index server. The whole system is files you could read with `cat` in ten years, which is the only durability guarantee that has ever actually held.

The tree

Only `raw/`, `INDEX.md`, `meta.json`, `proposals/` and `usage/` are mandatory. Everything else appears when you actually want it — an empty `glossary.md` nobody fills for six months is worse than not having one.

```
<your-os>/
├── INDEX.md          # scan-first entry point
├── config.json       # the authority on shape
├── meta.json         # counts, cutoffs, history
├── README.md         # this OS in your words
├── profile.md        # role, mandate, horizon
├── raw/              # append-only. source of truth
│   └── 2026-09-02--granola--acme-qbr.md
├── proposals/        # the gate, persisted
├── claims/           # INDEX.md + <topic>.md
├── jobs/             # INDEX.md + <job>.md
├── decisions/        # the forks still open
├── company/  glossary.md
├── connectors.md     # sources, protocols, blind spots
├── dashboards.md     # what is published, and from what
├── design.md         # which design system governs output
├── patterns/         # portable output specs
├── sensitive.md      # quarantine, out of the scan path
├── scripts/          # your own copies. 9 of them
└── usage/            # log.md, proposals.md
```

The two record shapes

A raw file is what was said. A claim is what you concluded, and every field on it exists to answer *should I still believe this*.

```
--- raw/2026-09-02--granola--acme-qbr.md
source: granola
person: dana-chen
date: 2026-09-02
type: meeting
jobs: [job-004]
external_id: granola:abc123   # dedupe
processed: false           # the one edit
```

```
### CL-0042 — Acme flagged pricing as their top…
- Statement: the economic buyer named per-seat…
- Kind: fact        # fact|decision|theme|
                        # assumption|constraint|metric|preference
- Jobs: job-004
- Confidence: confirmed
- Source fidelity: verbatim   # the medium
- Sensitivity: internal
- Source: raw/2026-09-02--granola--acme-qbr.md
- Citation: "the seat math is the only thing
  my CFO is going to push back on."
- Verified: 2026-09-02
- Decay: 90d        # or none, for durable facts
- Relations: supersedes CL-0031
```

WHY FIDELITY IS SEPARATE FROM CONFIDENCE

One enum was carrying three independent questions: how faithful is the recording, how many sources exist, and is the claim contested. Measured in a real corpus, **599 of 836 claims** sat one rung below the top because they were single-source summaries — and their transcripts were still fetchable through the same connector that produced them. One API call. **Zero had taken it**, because a ceiling that is elective looked exactly like one that is permanent. Splitting the medium out turns that into a sortable backlog of entries one call from being stronger.

How data is retrieved

The scan contract

Every skill reads in this order and stops as soon as it has enough. The corpus is never loaded wholesale — `corp-os-rebuild` is the single exception, and re-deriving everything is its entire job.

The ladder

1 **`INDEX.md`**
Counts, what changed, the cross-cutting views. Often the whole answer.

2 **`jobs/INDEX.md`**
One line per job: the statement, and what it still needs to know.

3 **`claims/INDEX.md`**
One line per claim, with its confidence and date visible without opening anything.

4 **`the detail file`**
Only now, and only the ones the index pointed at.

× **`raw/`**
Never wholesale. It is the archive, not the working set.

This is why an entry that exists as a file but is missing from its index is a **broken contract** rather than a small omission: the file is invisible to every skill that obeys the ladder.

What comes back, and with what attached

An answer, cited. `corp-os-recall` returns the claim, its confidence, who said it and when — so you can tell a confirmed quote from a reconstruction at a glance rather than trusting the summary.

A view, rebuilt. A dashboard is produced by a script that recounts what is on disk, not by a model writing numbers into markup. A hand-written dashboard had wrong counts within a day; that is the same failure `build_index.py` exists to prevent, one directory over.

A gap, stated. If a pattern's requirement does not resolve, the panel is omitted and said out loud. An empty panel reads as a state rather than a defect, and nobody investigates it.

Nothing sensitive, by accident. Load-bearing sensitive material is in the scan path on purpose, so anything rendered for a screen ships redacted and reveals by script — never the reverse, which is visible the moment the script does not run.

The five things that are not configurable

Everything else is yours

Layer names, vocabulary, decay windows, retention, gate strictness, and the organizing idea itself are all declared in `config.json` and meant to be changed. These five are not, because everything else rests on them.

01
An append-only source layer
Existence in it means *said*, not *true*. Lose this and nothing downstream can be checked against anything.

02
Provenance on derived entries
Source, date, verbatim citation, confidence. A paraphrase in the citation field defeats the purpose; a fabricated one is the only thing here treated as corrosive.

03
A review gate
In front of anything treated as knowledge, persisted to disk rather than held in a conversation.

04

An index that can be scanned

Rather than a corpus that must be loaded. One line per entry, always a descriptor.

05

Something that removes things

The one most systems lack, and the one that decides whether yours is still worth opening in two years.

AND ONE RULE ABOUT THE MACHINERY

A step that has to happen every time, that nothing else will catch if it is skipped, belongs in code rather than in an instruction. That is why there are eleven scripts and not eleven more paragraphs. Each was prose first, each held at a stubborn rate through two or more rewrites — the redaction log appeared under four different names across five runs and once not at all — and each only moved when it stopped being something to remember. None of them exercises judgment; that stays with the skill, and with you.

---

corp-os v0.14.1 23 skills · 8 commands · 11 scripts · 10 reference specs · 3 fixtures

5 invariants · 3 layer roles · 1 operator