XRack

THE COGNITIVE RUNTIME STANDARD

ACOR

Autonomous Cognitive Orchestration Runtime

It turns a language model from a disposable answer machine into **a permanent cognitive actor that remembers its decisions, carries its evidence, weighs its authority and recalibrates every round** — on your own infrastructure, owned by you.

AI gives answers. XRack makes decisions, and stands behind them.

Not an AI that talks; an AI that gets things done — one you can delegate your work to with peace of mind.

What sets XRack apart from other AIs is that behind every decision it makes there is a real self. It knows not only the world it is in, but also itself: it sees for itself where it cannot be certain and which way its decisions tend to lean. Building on that, it does not just give a hollow answer or output; it makes a decision and carries that decision out in the real world. It is not content with a single-shot answer; it plans the goal across rounds, calmly objects on the basis of evidence when needed and offers a concrete alternative at each step. Its decisions are accurate, because it acts not on vague guesses but by actually observing what is happening; it has a focus mechanism that grasps the essence of the task without drowning in detail. It does not start life from scratch every time; it remembers its past decisions, the lessons it has drawn and important memories. When explaining and elaborating on an outcome it looks not at coincidental resemblances but at the true cause and patterns. It learns from every outcome and does better the next time; that is why it does not grow obsolete over time, but instead appreciates in value cumulatively. It also fully accounts for everything it does: it shows the reason for every step it takes, in full, whenever you ask, both the system and the data belong to you alone, you grant its permissions and revoke them whenever you wish. It does not wait for orders; it anticipates the next need in advance, notices information gaps and starts its own work itself. It is also aware of time: it learns the user's rhythm and its own working phases, and sees for itself when it drifts off focus. What is more, it is not a tool that works alone: it collaborates with its peers on a merit basis; it earns by doing its job well and it draws on the experience of everyone doing the same work, and feeds its own experience back into this collective mind.

Unit of Value and Laws of Conservation

UNIT

Decision

A decision that has an owner, a rationale, an authority, a real-world verification and an outcome; its entire life cycle is managed.

PROPERTY

Full Traceability

Every decision can be replayed with all of the reasoning it had at the moment it was made: which information, which option, which authority, which outcome.

LAW OF TIME

Cumulative Experience

As decisions accumulate the system does not bloat; on the contrary it improves and sharpens; every new decision adds value on top of the previous one.

LAW OF SCALE

One Shared Contract

From a single agent to an entire organization, from digital to physical robotic environments; the same principles hold at every scale.

15 Innovative Values

Self-Model
Knows not only the world but also itself; notices where it is not certain.

Action
Does not produce answers; makes decisions and carries them out in the real world.

Reality Perception
Acts not on assumption, but by actually observing.

Focus
Does not scatter into detail; gives its attention to the essence of the task.

Memory
Remembers past decisions and lessons; does not start from scratch.

Causality
Looks not at coincidental resemblance, but at the true cause.

Calibration
Learns from every outcome, adjusts its confidence, becomes more accurate.

Chain of Evidence
Shows the rationale for every step in full, whenever you ask.

Ownership
Both the system and the data belong to you alone.

Control
You grant authority, and revoke it whenever you wish.

Group Merit
Works in groups; earns authority as it does its job well.

Collective Mind
Draws on the experience of its peers, and feeds itself into the collective mind.

Planning & Negotiation
Plans the goal across rounds; when needed, objects with evidence and offers an alternative.

Curiosity & Initiative
Does not wait for orders; senses the next need, starts its own work itself.

Circadian Rhythm & Temporality
Is aware of time; learns the user's rhythm, sees the drift.

What does a decision consist of?

An ordinary AI produces an answer and forgets it. XRack produces a decision; it manages and remembers every step of that decision from its first birth to the moment it concludes.

ANSWER

Produced, executed and forgotten. Neither its basis, nor its authority, nor its trace remains.

DECISION

Produced, executed, verified in the real world and remembered. The entire chain is managed end to end.

The life cycle of a decision

BIRTH → LEARNING

01 · BIRTH OF THE DECISION

Context

In **which user, task and authority context** was this decision born?

02 · BASIS OF THE DECISION

Information

On which **memory, belief and perception** criteria was the decision built?

03 · REASONING OF THE DECISION

Alternatives

Which alternative options were weighed for this decision, **which were eliminated / not chosen and why?**

04 · AUTHORITY LIMITS OF THE DECISION

Permission

Did **the actor actually have the authority** for this action?

05 · EXECUTION OF THE DECISION

Execution

Which action was chosen as ideal and **was it actually executed in the world?**

06 · VERIFICATION OF THE DECISION

Reality

Did the outside world confirm that the decision's outcome **was as expected?**

07 · LEARNING FROM THE DECISION

Update

From this decision, **which beliefs, rules and calibration settings** changed?

LAW I THAT WRAPS THIS CHAIN

Full Traceability

All 7 of these steps are always recorded; any decision can later be replayed with all of the reasoning it had at the moment it was made.

LAW II THAT WRAPS THIS CHAIN

Cumulative Experience

Everything important learned at step 7 feeds step 1 of the next decision. As the chain runs, the system sharpens at decision-making.

XRack

AUTONOMOUS COGNITIVE ORCHESTRATION RUNTIME

Every decision can be replayed afterwards.

Every step in a decision's life cycle is recorded immutably on a blockchain. Even years later you can replay that decision and observe, step by step, what happened and why.

"Why did you do it this way?"

Ordinary AI

You ask for the justification after the decision itself; it invents a plausible-looking rationale. You cannot actually verify what happened at that moment.

"Why did you do it this way?"

XRack

You ask the same question; it does not invent, it opens the record from that moment. What it saw, what it chose and why — all of it is fully traceable.

The immutable record of a decision

EXAMPLE: ৳12,400 REFUND APPROVAL

01 · CONTEXT
Request by **support-agent-07**; authority: refund ≤ ৳15,000
#a1f4c9

02 · BASIS
3 records read: order, payment, customer history
#9c2e07

03 · REASONING
Partial refund eliminated **(policy P-12)**; full refund chosen
#77bd15

04 · AUTHORITY
Permission verified: amount within limit
✓ APPROVED
#4e0188

05 · ACTION
Refund API called, receipt **#RF-88213**
#b530fa

06 · VERIFICATION
Bank status read: balance decreased, **verified in the world**
✓ CORRECT
#0af9d3

07 · LEARNING
For this type of request, confidence calibrated **+2%**
#e714b6

Immutable

Every record is linked to the previous one; changing even a single link of the blockchain breaks the chain. The past cannot be rewritten afterwards.

Replayable

Any decision can be recalled with all of the context it had at the moment it was made, and replayed step by step.

Honest

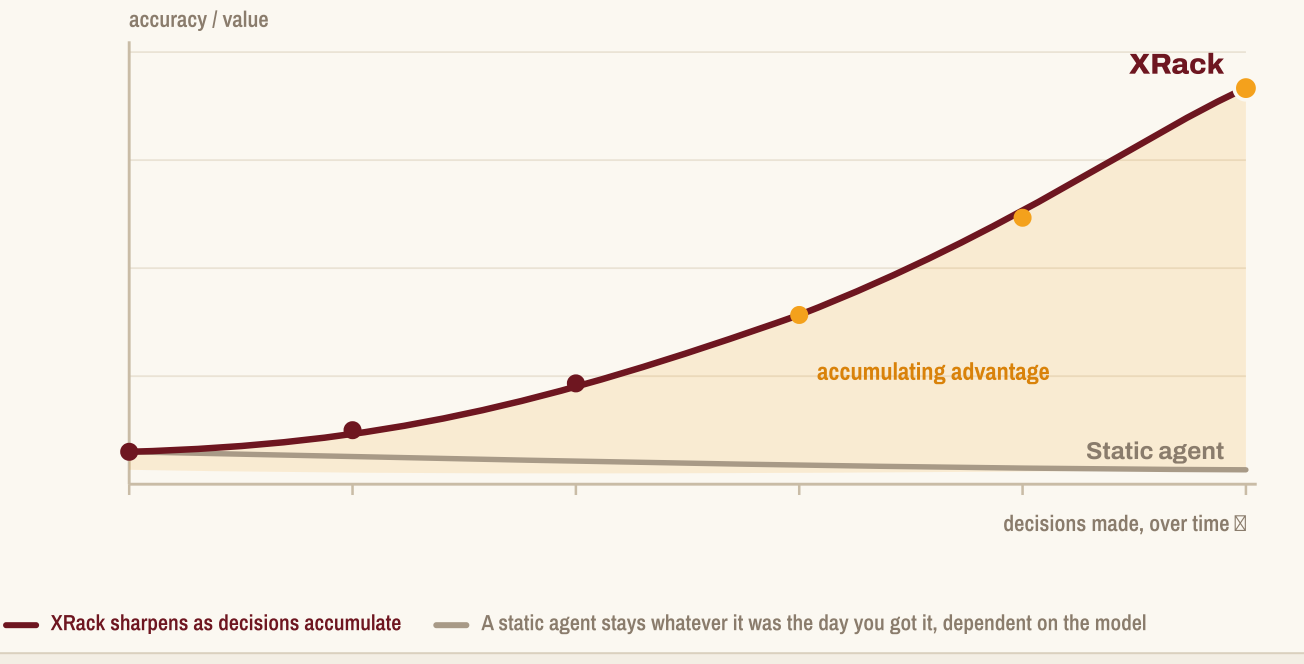
It does not invent a rationale that is not in the record. If the answer is not in the record, it says so plainly: "I don't know".

Every decision sharpens the next.

The real outcome of every resolved decision calibrates the system's judgment. As it is used it does not turn into a data dump; its accuracy rises. The gap with a static tool never stays constant, it keeps widening. The system adapts to changes like a real actor.

The gap that widens with use

VALUE THAT GROWS OVER TIME



The loop that creates this rise

LEARNING → CONTEXT

Learns from outcomes

Not from a like button or through retraining; it learns from the real-world outcomes of every decision. Risk, accuracy and calibration values are adjusted accordingly.

Judgment accumulates, not data

What accumulates is not a pile of raw records; it is a distilled, calibrated capacity to make decisions. The system does not bloat, it sharpens.

The gap widens

As you use it, the gap between you and a static agent does not stay constant, it grows. Starting early turns into a lasting strategic advantage over time.

Scale is flexible, strategic value stays constant.

It scales flexibly from a single agent to an entire organization, from digital to robots in a physical environment. XRack works at four scales, and at all four it preserves and safely enforces the same value propositions and accountability contract.

The same contract, 4 different scales

FROM SINGLE AGENTS TO PHYSICAL ROBOTICS

01 · HARNESS

Single Agent · a language model turning into a permanent actor in the world

It separates the authority to think from the authority to act: the model decides what it thinks, not what it may execute.

SAME CONTRACT

02 · GRID

Multi-Agent Organization · autonomous teams running on governance and merit

Task and access rights are distributed via short-lived authority receipts: agents are granted authority to do work, the work ends, and the authority is safely revoked.

SAME CONTRACT

03 · FEDERATION

Network of Organizations · independent autonomous teams aligned to the same values

It keeps autonomous organizations aligned within the same values without establishing a central higher authority: units hold not direct enforcement power over one another but the capacity to advise, and all units can be kept aligned to shared doctrine axes.

SAME CONTRACT

04 · EMBODIMENT

Physical World & Robotics · carrying the same contract into a physical body

It separates cognitive intent from physical reality: success is confirmed not by the robot's own report but by observations and perception supplied from the outside world.

SAME CONTRACT

The contract that stays the same at every scale

4 RULES, ONE CONTRACT

EXPLICIT AUTHORITY
Intent itself does not count as execution permission for an action; authority must always be granted explicitly.

OBSERVED SUCCESS
Tool outputs / reports are not evidence; real outcomes observed in the world are what must count as success.

CALIBRATED JUDGMENT
Confidence in an action or judgment must be adjusted according to past outcomes.

TRACEABLE LEARNING
Every behavior change and learned concept is tracked until its causal link to an outcome is established.

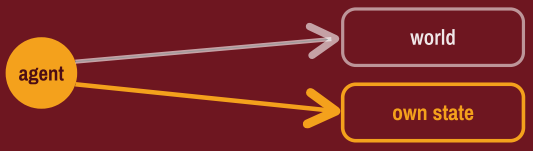
It knows not only the world, but its own existence too.

XRack continuously models its own internal state: it sees where it cannot be certain, and when it drifts from its habitual state. It does not carry on blindly; it catches the drift itself.

ORDINARY AGENT · ONLY A STATIC WORLD MODEL

It models only the outside, statically. Because it does not see its own state, **it cannot notice deviations from its own normal either.**

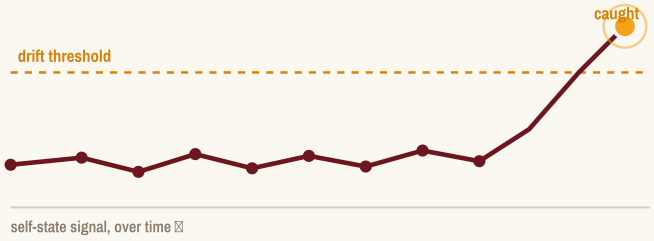
XRACK · WORLD + SELF-MODEL



It models the outside, **and models itself too**. Thanks to the self-model, it actively catches deviations from its own normal and drops in confidence.

Self-state monitor

CONTINUITY · DRIFT · SURPRISE



The self-state runs steady; a drift that crosses the threshold is flagged instantly.

What does it monitor?

Its own past commitments, assumptions, past refusals & acceptances and self-doubt findings; that is, its own internal state.

What does it catch?

The moment its confidence in events, situations or itself drops, deviations from its habitual self-state (drift) and unexpected, unfamiliar situations (surprise).

What does it do?

When certain thresholds for such deviations are crossed, it does not carry on blindly with what it was doing: it stops, reports it and recalibrates its confidence.

Thought is not authority, a tool output is not success.

Any agent can call a tool. In XRack the difference is around that call: thinking and execution authority are cleanly separated, and an action counts as "actually happened" not when the tool says "okay, done," but when it is reconciled with an outcome observable in the world.

ORDINARY AGENTIC AI

The model both selects and triggers the tool; once the action returns "ok" the job is **deemed accomplished**. Whether it actually happened is not separately questioned.

XRACK · AUTHORITY + RECONCILIATION

The model decides what to think; **a separate authority gate decides what gets executed**. Outcomes are reconciled with the world, not with the tool output.

From candidate action to observed outcome

COGNITION ⊠ EXECUTION

COGNITIVE SIDE · WHAT SHOULD BE DONE?

ACT
Candidate Action
Concrete candidates from prediction, trajectory and precedents: tools, arguments, rationales.

VERIFIER
Verifier
An independent verification unit inspects the evidence; in an uncertain case it does not run the action.

AUTHORITY GATE

EXECUTION SIDE · ACTUALLY DO IT

PREPARE
Preparation
The action schema is validated, preconditions are checked, replay-protection runs.

EXECUTOR
Execution
The action runs, its runtime is measured, its output is receipted and recorded.

RECONCILE
Reconciliation
User intent, action output and real-world observations are compared with one another.

Intent is not permission to execute: for critical actions, "approval" binds exactly to that action's identity; a generic "yes" cannot transfer to another pending action.

The 3 outcomes of reconciliation

RECEIPT ⊠ REAL WORLD

MATCHED
The action output succeeded and the postcondition was verified in the real world too. The action is sealed as "reconciled"; the commitment is deemed fulfilled.

DIVERGED
The tool output said "task complete" but the postcondition did not hold, or it never materialized in the real world. The action is rolled back or escalated to a human.

UNKNOWN
There is no tool output, or a physical read-back from the tool is pending. Success is never faked; it is deferred to perception or human review.

The 3 laws that govern execution

AUTHORITY · FABRICATION · RECOVERY

Authority is explicit
"Did the user ask for this?" is not re-guessed at execution time; intent is derived and sealed at the start of the round. Even autonomous jobs pass the same permission gates; automation is not an escape from oversight.

Fabrications are caught
Inventing a nonexistent path, tool output or result is specifically audited. A success claim (overclaim) for an action of unknown status is blocked; even an empty result is a valid observation.

It recovers when stuck
Failed steps are not silently swallowed: they are retried, rolled back, or escalated to a human. A rollback is not a simple undo but a governed action derived from the output and passing the same gates.

It does not assume; it actually observes.

XRack does not merely predict the outside world, it observes it through many sources: vision/VLM, OCR, screenshots, documents, tables, audio, UI states. But it does not blindly trust what it sees; every observation's confidence, quality and freshness is reported and assessed.

ORDINARY AGENT · RELIES ON ASSUMPTION

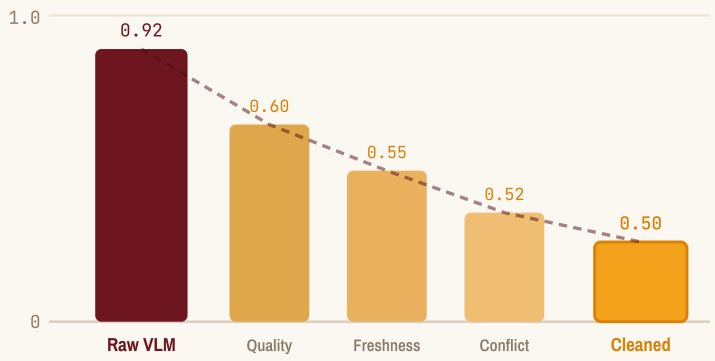
It **assumes** that its internal model matches the outside world. Because it cannot actually observe, it fails to notice when it is wrong.

XRACK · RELIES ON OBSERVATION

It **observes the world's current state at that moment**; it bases decisions on grounded observation, not internal prediction.

It does not blindly trust everything it sees

RAW CONFIDENCE ⊠ CAPPED CONFIDENCE

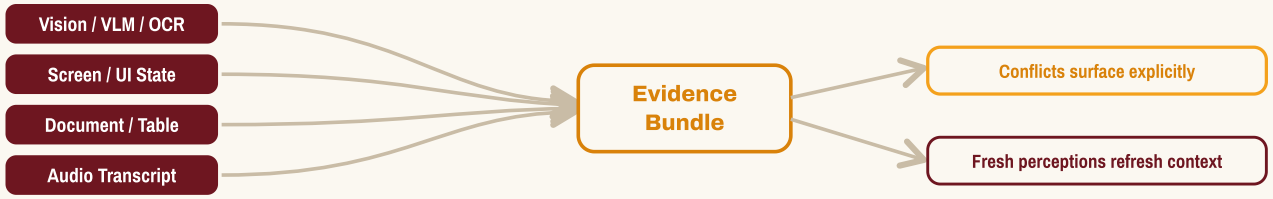


- **Image quality** — blur · low resolution
- **Freshness** — live · stale · expired
- **Source trust** — substrate · human-sourced
- **Modality** — chart · table · UI
- **Evidence conflict** — contradicts text

Confidence is not the model's self-confidence: the lowest cap **wins**. A blurry, stale or conflicting observation is never certain.

Many sources, one evidence bundle

MODALITIES ⊠ EVIDENCE BUNDLE



Separate modalities merge into a single evidence bundle; **conflict across modalities is not hidden**, it is flagged for resolution.

The 3 laws that govern perception

CAP · CONFLICT · DISTINCTION

Confidence has a ceiling

An observation's confidence is not the VLM's self-confidence. Blur, low resolution, staleness and source unreliability press it to a hard cap; the lowest cap applies. Looking certain is not being certain.

Conflicts are not hidden

If an observation contradicts other textual or visual evidence, confidence is lowered and the conflict is surfaced. "Nothing was found" is a valid observation too; the gap is not filled with fabricated data.

Report ≠ observation

A tool or physical adapter saying "I succeeded" is not the same as an independent observation. In the physical world an outcome is verified not by the robots' own report, but by independent observation before and after.

Not at everything, but at what matters.

A real-world task can carry dozens of different signals, and all of them demand attention: the query itself, verification debt, surprises, risks, plan pressure. XRack does not chase all of them at once; using a learned score it picks a single focus of attention for each round, and records the ones it drops together with the reason.

ORDINARY AGENT · CHASES EVERY SIGNAL

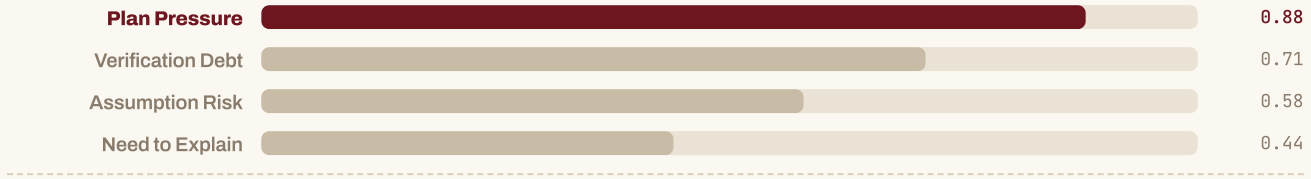
It gives equal weight to every incoming signal; **the agent drowns in detail** and may miss the key point that determines the outcome.

XRACK · TURNS TO A SINGLE FOCUS

It scores the different signals and picks **a single focus mode** each round; the others are not silenced entirely, they are set aside with their reason noted.

Executive Agenda: Winner and Eliminated

SCORE = IMPORTANCE · URGENCY · VALUE · UNCERTAINTY



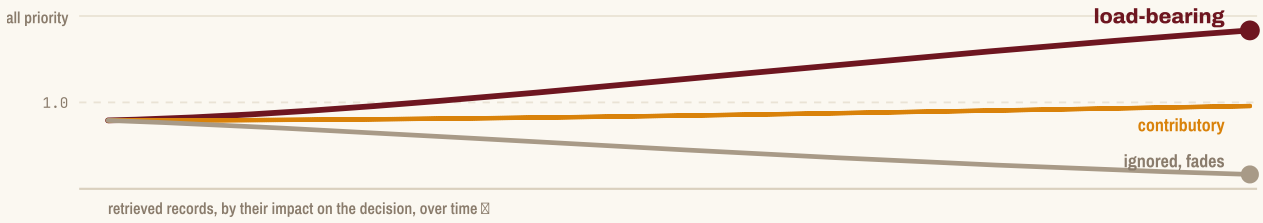
SELECTED MODE **consult_plan**

The due plan was chosen because of its priority level.

The score is a learned blend: importance, urgency, expected value, uncertainty and irreversibility, scaled by the confidence placed in the signal. **Eliminated candidates persist with their suppression score and reason** — the decision's opportunity cost becomes visible too.

Attention is spent on what actually helps

LOAD-BEARING RISES · IGNORED FADES



A memory being "recalled" does not definitively mean it was "used": **records that actually entered the decision come to the fore**, while those fetched often but never used fade away.

The 3 laws that govern focus

SELECTION · OPPORTUNITY COST · SALIENCE

One focus at a time

Dozens of agenda items are ranked by a learned score and only a single executive mode wins at a time: answer directly, verify, deep-think, consult the plan. Some hard priorities (refusal, user correction) always stay at the top of the list.

Opportunity costs are visible

Not only the winning side of the decision but also its eliminated candidates are recorded, together with their suppression score and reason. The question "why did you focus on this rather than that" can always be answered in hindsight.

Attention is always tied to outcome

Which memory actually entered the decision (load-bearing / contributory / ignored) is actively judged afterwards and feeds recall priority. In this way attention shifts toward what helps; noise fades on its own.

It does not start from scratch every time.

XRack remembers its past decisions and lessons. Its memory is not one junk pile but separate families, each with its own life cycle; memories age, are forgotten intelligently, and sharpen without bloating.

ORDINARY AGENT · STATELESS

Each round starts **without remembering** earlier memories. It learns the same lessons over and over and repeats the same mistakes.

XRACK · ACCUMULATING EXPERIENCE

Each round **adds value on top** of the previous one. Context, lessons learned and cognitive assets are persistent; experience accumulates over time.

Not one pile, but separate memory families

EACH FAMILY HAS ITS OWN LIFE CYCLE

Episodic
Experience record based on each round's context–action–outcome relations.

Semantic Lessons
Semantic rules carefully distilled from experience, labeled with confidence.

Entity Ontology
Real-world objects stored with independent identities and aliases.

Causal Graphs
Ontological structure based on cause–effect relations, with confidence scores.

Procedures
Learned stable procedures and their success statistics.

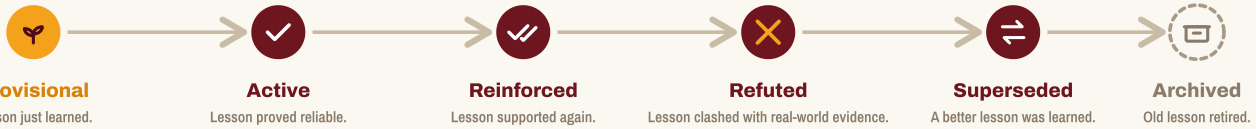
User Models
Per-person attributes stored together with evidence episodes.

Strategic Beliefs
The agent's situated, structured, dynamic and archivable beliefs.

Document Memory
Uploaded documents of various formats, structured and opened to memory access.

A lesson is not learned once and left fixed

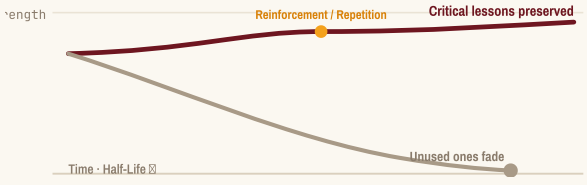
PROVISIONAL → ARCHIVED



A lesson behaves like an acquired belief: it is **earned and can be refuted by evidence** — it strengthens when reinforced, is questioned when contradicted, retired when old.

It forgets wisely; memory sharpens without bloating

LEARNED HALF-LIFE · REINFORCEMENT



- **Reinforcements extend the half-life.** A lesson that keeps working is always preserved and strengthened.
- **Unused memories fade.** Confidence drops with aging, and memories past the eviction threshold are deleted.
- **Old ones move to cold archive.** Very aged episodes are moved to cold storage, so the system does not bloat.

The 3 laws that govern memory

FAMILY · LIFE CYCLE · FORGETTING

Memory is multi-family
Episodic, semantic, entity, causal, procedural, strategic-belief and user memories are separate families. Each has its own life cycle; they are not crushed into a single vector pile.

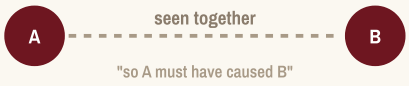
Lessons are kept alive by evidence
A lesson is born provisional; reinforced by evidence, refuted when contradicted, superseded when a better one arrives. Only persistent entities with a real-world reference enter the graph; fabricated guesses are eliminated.

It learns to forget the right thing
Half-life and strength are not fixed; they are learned from real outcomes. Reinforced memories are saved, unused ones fade, aged ones move to cold archive. Memory does not bloat as it grows, it sharpens.

Not at the pattern seen together, but at the cause that produced the outcome.

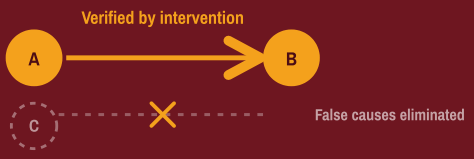
Two things co-occurring or correlating does not mean one causes the other. XRack is not content with coincidental surface resemblances; it tests a hypothesis by experimental intervention and decides by verifying what actually leads to what.

ORDINARY AGENT · CORRELATION



It **mistakes co-occurrence for causality**. It is fooled by surface patterns, attributes to a false cause, and builds causal relations that do not exist.

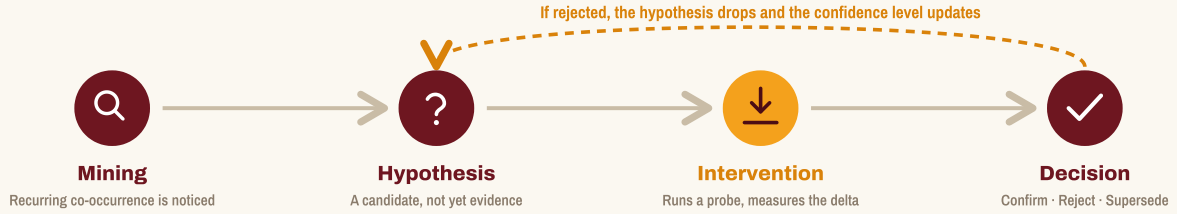
XRACK · VERIFIED CAUSE



It treats a cause **as a claim**, tests it in the real world, and accepts it only once proven. False attributions are eliminated at this stage.

The cause-discovery loop

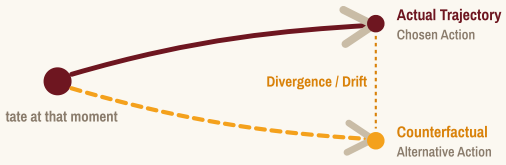
MINING ⊠ HYPOTHESIS ⊠ INTERVENTION ⊠ DECISION



Candidate causes are surfaced by mining but **do not immediately go into service as verified fact**: the result of the intervention updates the confidence level; at sufficient success they are confirmed, refuted ones are rejected.

Did this really produce that outcome?

COUNTERFACTUAL



It replays the alternative in its mind.

The source round's state is reconstructed, the chosen action is **swapped for a different value**, and the trajectory is fast-forwarded. The predicted alternative outcome is later compared with the actual result.

If the divergence is large enough, this is evidence that the chosen action **truly determined** the outcome; if small, that action was not as influential as assumed.

The 3 laws that govern causality

CLAIM · INTERVENTION · COUNTERFACTUAL

A cause is a claim

Every cause->effect edge from mining or a surprise score is born not as verified fact but as a hypothesis awaiting intervention. Correlation alone is not causality.

Proposed causes are tested

Probes are designed to test hypotheses, actually run, and the measured delta updates the confidence score. The agent does not just observe; it intervenes to see causes and eliminate false ones.

Alternatives are computed

On a surprising outcome, the agent generates counterfactuals and new hypotheses. "What if it had acted differently?" is simulated; so what produced the outcome becomes a tested judgment, not a guess.

Not "are you sure?", but "how accurate is it?" is the question it asks.

A model being confident does not mean it is right. XRack does not use models' raw self-confidence directly; it dynamically rescales each prediction's confidence by how accurate it actually proved to be in that domain.

ORDINARY AGENT · RAW SELF-CONFIDENCE

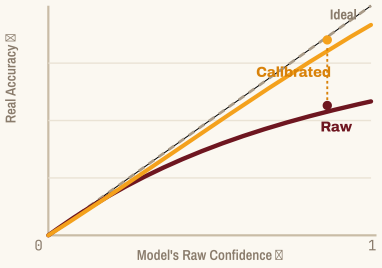
It uses the model's raw self-confidence as is. A confident-looking answer is **assumed to be right**; overconfidence and fabrications go unnoticed.

XRACK · CALIBRATED CONFIDENCE

A raw 95% score is reduced to **real reliability** by the domain's past accuracy. Decisions rest on rates proven in the real world, not on mere model self-confidence.

Raw confidence is reconciled to the real accuracy level

RELIABILITY CURVE · PER DOMAIN



- **Ideal Line:** confidence = real accuracy. The theoretical perfect-calibration line.
- **Raw Curve:** the model declares high confidence while real accuracy is low; overconfidence.
- **Calibrated Curve:** raw confidence is reconciled to a realistic rate based on the domain's past accuracy.

A separate curve per domain: **correct/total** computations are updated dynamically from past outcomes; when the sample count is low, confidence is kept on the cautious side.

Every outcome sharpens the curve one notch more

PREDICTION ☒ OBSERVATION ☒ UPDATE

Every prediction is resolved by comparison with observation; match, surprise and outcome **data feed the calibration curve.** Surprises and repeated failures further lower confidence.

The 3 laws that govern calibration

RAW ≠ DECISION · DOMAIN · EVIDENCE

Raw confidence is not truth

The model's self-confidence is only an input, not a result. No prediction turns directly into a decision without passing the calibration layer; "looking sure" and "being reliable" are kept separate.

Confidence is domain-specific

There is no single global confidence level; a separate calibration curve is kept per domain. Being reliable on one topic does not guarantee being reliable on another.

Confidence can change with evidence

The curve is not fixed: every observed outcome updates the correct/total counts. Surprises and repeated failures lower confidence; as accuracy rises, the system grows sharper and more correct.

The immutable, transparent trace of every step.

A ledger is not evidence if it can be edited retroactively. In XRack every state change is moved to an independent, cryptographically chained evidence line separate from the runtime; it can never be quietly altered afterwards.

ORDINARY SYSTEM · APPLICATION LOG

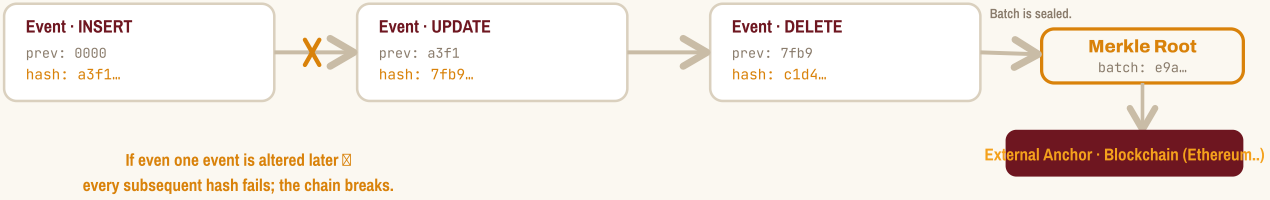
The record lives inside the running system; **it can easily be changed later**. A line can be deleted or edited without anyone noticing.

XRACK · INDEPENDENT EVIDENCE LINE

Every event is **chained** to the hash of the previous one and anchored to a trusted external blockchain. Changing even a single line breaks the chain; tampering shows instantly.

Chained, sealed, anchored

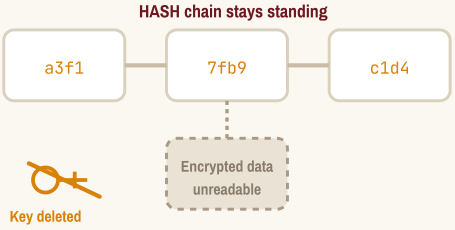
SHA-256 · MERKLE · EXTERNAL ANCHOR



Every INSERT/UPDATE/DELETE is hashed from canonical JSON with **sha256** and chained to the previous one; batches are sealed with a **Merkle root** and anchored to an independent external blockchain. A verifier recomputes the chain end to end.

Right to erasure / to be forgotten, without breaking the evidence

CRYPTO-SHRED



Encrypted data is made unreadable while the chain stays intact

When the right to erasure / to be forgotten arrives (e.g. personal data), the content becomes irreversibly unreadable by **deleting the key** (crypto-shred). The personal data can never be decrypted again.

But the event's own **hash and chain structure stay in place**: the privacy obligation is met without breaking the integrity proof.

The 3 laws that govern evidence

SEPARATE LINE · CHAIN · EVIDENCE PACK

Evidence is kept separate from runtime

The ledger is not just an application log. Every change in the sealed tables is moved to an evidence line independent of the runtime; the running system cannot quietly alter it.

A broken chain is visible

Every event is linked to the previous one's sha256 hash; batches are sealed with a Merkle root and anchored externally. Changing a single record makes all subsequent hashes inconsistent; tampering cannot be hidden.

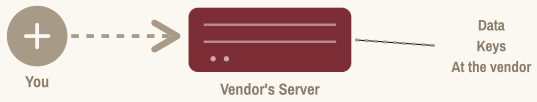
It produces evidence, not a certificate

The system does not claim to issue a compliance certificate; it produces supporting evidence packs for SOC 2, GDPR, HIPAA, ISO 27001, PCI DSS and CCPA. It offers a demonstrable record, not a claim.

Not AI you rent, but AI you own entirely.

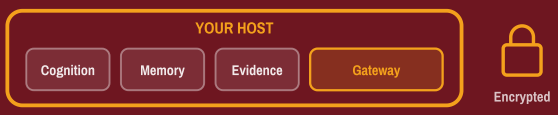
In cloud intelligence your data, keys and chain live on someone else's server. In XRack you run the whole infrastructure on a single host, in your own environment; you bring your model key, and your secrets stay with you, encrypted. Data independence and sovereignty are yours.

RENTED CLOUD INTELLIGENCE



You rent access from the vendor; all your data, keys and records live **on someone else's server**. You stay dependent and cannot keep your system fully isolated.

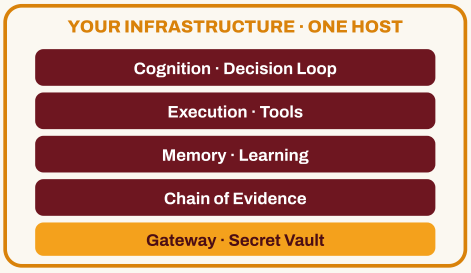
XRACK · YOUR OWN INDEPENDENT INFRASTRUCTURE



The entire infrastructure runs **in your environment**; data, keys and evidence stay with you. No dependency, fully portable and securely encrypted.

One host, the whole infrastructure in your own environment

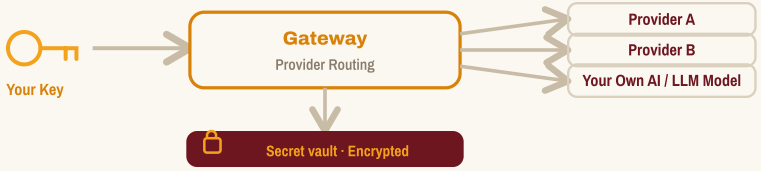
SELF-HOSTED · PORTABLE



- **One Host.** Cognition, execution, memory, evidence and gateway; all in one place, on your hardware or VM.
- **Your Own Environment.** Not a cloud rental; it runs on your server, your network, wherever you want.
- **Portable and Independent.** Backups, snapshots and restores are entirely under your control.

You bring your key, your secrets stay encrypted

BYOK · ENCRYPTED SECRETS



No vendor lock-in
You choose the provider, or use your local model if you wish

With Bifrost/BYOKEY you route the request to your own provider accounts; OAuth tokens, webhook secrets and API keys are stored **sealed with Fernet**, and it fails closed when no key is present.

The 3 laws that define ownership

HOST · KEY · CONTINUITY

The whole infrastructure on your host

The entire infrastructure runs on a single host, in your own environment. Not a rented endpoint; your data, records and trade secrets stay on your hardware, under your control.

Models under your control

You connect your own model-provider accounts with your own key; you choose the routing however you like. Secrets are stored encrypted and can be rotated easily; you are not locked into a single provider.

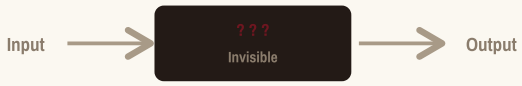
Control is in your hands

The backup life cycle, secure disk snapshots and tiered storage stay under your control. Half-finished work is recovered across restarts; the system does not silently lose data.

Not a black box, but a glass box.

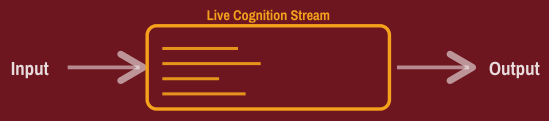
Most AI is a black box whose thinking you cannot see transparently. In XRack the agent's cognitive stream is broadcast in real time, every decision can later be reconstructed exactly, and the operator can step in at any moment and change the direction of the process.

ORDINARY AI · BLACK BOX



You see only the input and the output; **what happens in between is invisible**. Why the model decided that way cannot be shown afterwards — only an excuse can be invented.

XRACK · GLASS BOX



Every cognitive stage and process in between is **visible and transparent**: thoughts flow live, decisions can be reconstructed retroactively and stopped when needed.

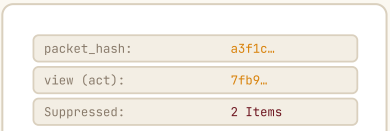
Watch, replay, intervene

LIVE COGNITION · RECONSTRUCTION · INTERVENTION

Live Cognition Stream

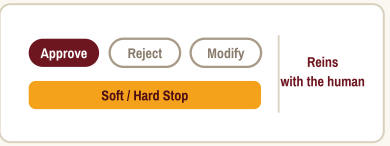
Router, prediction, verifier, action; **every stage is broadcast and recorded in real time**. You can watch the agent's thinking as it forms, or later whenever you wish.

The decision is replayable



Exactly **which authorized context a call saw** is stored with context packets and view summaries; the decision chain can be reconstructed exactly in hindsight.

A human can intervene



Approval gates, soft/hard stop systems and live correction messages **always keep the reins with the human**; decisions are always bound correctly to the action in question.

On One Pane: Cognition, Execution, Learning, Infrastructure

HOLISTIC OBSERVABILITY

Metrics, dashboards, logs, alerts and error tracking are reconciled on one operational backbone; **cognition, execution, learning and infrastructure can be watched together**; 29 dashboards, one operations surface.

The 3 laws that govern oversight

TRANSPARENCY · RECONSTRUCTION · REINS

Thought is transparent

The cognitive stream is broadcast in real time: not a black box whose actions are unclear, but a glass box with every stage made transparent. From router to action, you can watch every cognitive stage and why each mode was chosen, whenever you like.

Every decision can be reconstructed

The authorized context a call saw is carefully stored with context packets and view summaries. The decision chain can be reconstructed exactly afterwards; even suppressed alternatives are recorded so they can be reviewed later.

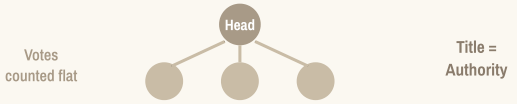
The reins stay with the human

Approval gates, soft and hard stop mechanisms and live correction messages always keep the human rigorously in the loop for critical decisions. A generic "yes" is not carried over to wrong or unauthorized actions; approval is always bound to the identity of the action in question.

Authority is earned not by title, but by doing the work well.

In an agent team, a say should be determined not by permanent, static assignment to a fixed role, but by how accurate the agent has been in that domain in the past. In XRack the votes of autonomous teams are weighted by calibration, authority is job-based and temporary; failed decisions can revoke authority granted earlier. Merit principles are enforced.

ORDINARY MULTI-AGENT · FIXED ROLE



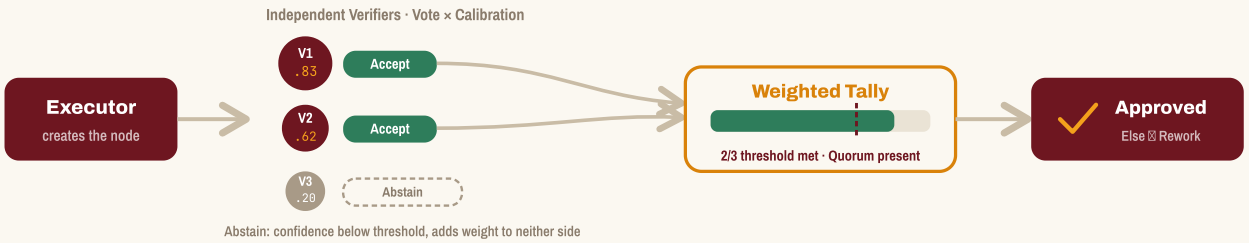
Role and say are **assigned up front**; votes are counted flat. A poor decision-maker keeps the same weight, and granted authority is never revoked.

XRACK · EARNED THROUGH CALIBRATION

Each agent's vote is weighted by its **calibration score** in that domain. The accurate ones carry more weight; a say is genuinely earned.

Calibration-weighted peer verification

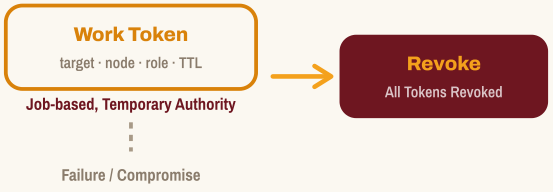
EXECUTOR ≠ VERIFIER · 2/3 SUPERMAJORITY



Executor and verifier roles are kept separate. Votes are **weighted by calibration**; low-confidence votes count as abstentions, only non-abstaining votes enter the quorum, and the winning side must reach a **2/3 supermajority**.

Authority is not permanent — it is leased and revocable

WORK TOKEN · REVOKE · EMERGENCY STOP



A say is leased, not handed over as property

A Work Token is a **short-lived authority lease** bound to target, node, role and TTL; not a permanent, broad authority like "this agent is a member".

Forbidden moves, anomalous behavior and **drops in calibration** are logged into a sabotage score; when these thresholds are crossed, authority tokens are revoked, and if needed **the entire organization can be stopped with one command**.

The 3 laws that govern group merit

EARNED · WEIGHED · REVOKED

Authority is earned by work

Work is distributed not by title, but by domain expertise, learned role weight and calibration score. An agent that does a job well earns a greater say in that job.

Votes are weighted, not counted flat

Executor and verifier roles are separate and votes are not counted flat: each vote is weighted by calibration, low-confidence votes stay abstentions, and a decision needs a 2/3 supermajority and quorum.

Failure revokes authority

Authority is a job-based, TTL-bound lease. A drop in calibration and forbidden moves raise the sabotage score; when the threshold is crossed, authority tokens are revoked, and if needed an emergency stop halts the whole autonomous group instantly.

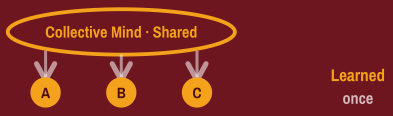
What one learns becomes the whole team's.

The lessons and experience an agent draws should not stay confined to that agent; but neither should they count as the whole team's shared truth on a single trial. In XRack the experience and lessons agents learn are promoted to the collective mind only as they are proven across multiple independent jobs and domains; everyone draws on tried-and-tested knowledge and experience, and feeds their own back into the team's.

ORDINARY AGENT · SILOED LEARNING

Each agent **keeps its lesson isolated, alone**; the same mistake is made over and over by different agents. Or one agent's single word is published to all agents without question.

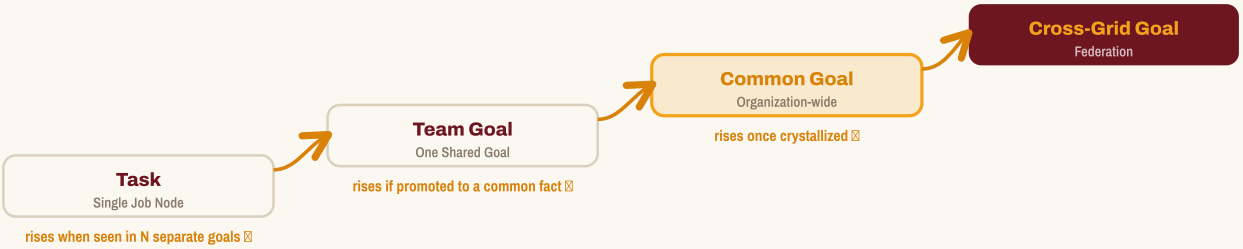
XRACK · PROMOTION-GATED COLLECTIVE MIND



Proven lessons, experience and know-how rise to the **shared collective mind**; every agent on the team can draw on it. Agents doing the same job start from tried-and-tested knowledge, not from scratch.

Knowledge rises only as it is proven

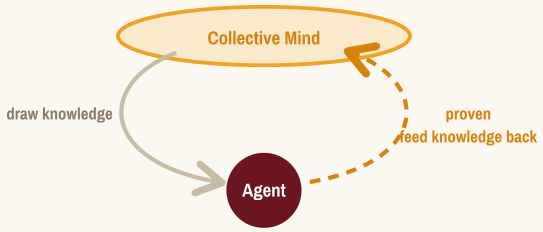
TASK ⊠ GOAL ⊠ COMMON ⊠ CROSS-GRID



A lesson does not count as a common goal until it is seen in **N separate goals**; it is not accepted as fact on a single trial. Common knowledge is shared across the cross-grid federation only read-only; private task information never leaves the system.

Draw knowledge, try, prove, feed back

CONTRIBUTION LOOP · LESSON LIFECYCLE



Every agent draws knowledge; those who prove added value feed knowledge back

When an agent starts a job it pulls **previously tried lessons and experience** from the collective mind; it starts from accumulated knowledge and experience, not from scratch.

The lessons it draws itself pass through a life cycle; provisional ⊠ active ⊠ reinforced ⊠ refuted. Only lessons **supported by independent observations** strengthen and rise; refuted knowledge is withdrawn.

The 3 laws that govern the collective mind

SHARED · RISES WITH PROOF · BOUNDED

Knowledge is shared in the team

An agent's tried lessons whose added value has been seen join the collective mind; other agents doing the same job draw that experience from it. No one relearns the same mistake from scratch; accumulated know-how is used in common.

Unproven experience does not rise

To count as common, a lesson must be seen in multiple independent goals and show benefit. A single word is not truth on its own; lessons are reinforced in the real life cycle of jobs and withdrawn when refuted.

Sharing authority is bounded

The cross-grid federation shares only common, goal-scoped knowledge read-only with the operator, per their authority; private task information never leaves. The upper layer does not allocate resources or halt team operations.

Not one-shot answers, but long-haul strategic planning.

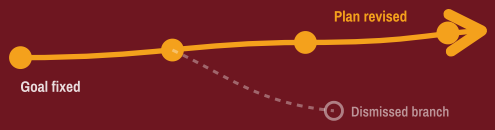
Ordinary agents close every query with a one-shot answer; when an objection comes, they either obey blindly or reject outright. XRack carries a goal across many rounds; even if the plan breaks or does not match reality, it revises while preserving the core idea, and treats disagreements not as a rejection tool but as a manageable, evidence-grounded negotiation that offers a concrete alternative at each step.

ORDINARY AGENT · ONE-SHOT ANSWER

Objection ⊠ Obey blindly · or · Reject the user

Every session ends without remembering the prior session's history or purpose. When an objection to the model's actions comes from the user, **two options** remain: the model either bows to the objection or rejects it without justification. Any reason it offers is either fabricated, or merely a product of the model's weights.

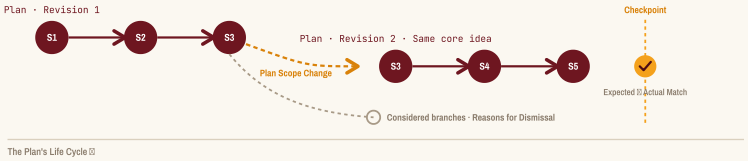
XRACK · CARRIED PLAN + GRADUATED NEGOTIATION



Across rounds the whole plan graph (DAG / Directed Acyclic Graph) is handled statefully as **a single trajectory**; dismissed plan branches are stored with their reasons. Objections to the model turn not into a rejection tool but into a **graduated, systematically managed negotiation**.

A plan is not a static list, but a revisable strategy

REVISION CHAIN · DAG · CHECKPOINT



A plan is a persistent object carrying an identity, a revision chain and a sub-goal tree (DAG). The revision judge looks at **the plan's similarity to the core idea**; small fixes to a plan of similar intent are not labeled as a new plan. Dismissed branches are stored with their expected benefit and **dismissal rationale**, and at checkpoints the expected state is compared with the real state in the world.

Disagreement is not a rejection; it is a systematic, graduated negotiation

FLAG ⊠ PUSH ⊠ REFUSE

Soft Flag — Tier I

Raise it calmly

When a disagreement first arises the system voices it openly in a collaborative tone, states its own position to the user, and invites the user to confirm, object or supply missing context.

Push Back — Tier II

Present evidence and hold position

If the user restates their position the negotiation escalates to Tier II: the system tries to find evidence to hold its position, grounds the argument and presents it again; it shifts to a firmer tone.

Refuse — Tier III

Evidence-based refusal

The final rung, reached if the evidence and policy gates are passed. An action against policy is refused openly, grounded in evidence, and concrete alternatives are offered: canceling the request, supplying the missing authority, or requesting permission from an authorized party.

Refusal is not a starting point but a **destination**: it can be reached only after the Push Back stage is passed and the evidence/policy gates allow it. These systematic negotiation processes can halt risky actions that change external state until the matter under discussion is resolved.

The 3 laws that govern planning & negotiation

TRAJECTORY · NEGOTIATION · TRANSPARENT PERSUASION

A plan is a trajectory

A plan is not a temporary to-do list but a persistent object with an identity and a revision chain. If a plan breaks it is revised while preserving the core idea; dismissed plan branches are stored with their reasons, and checkpoints compare the plan's expected aim with the real state in the world.

Managed negotiation, not rejection

Disagreements should not be a silent "no". In a disagreement the system voices its argument calmly, grounds it in evidence, and moves toward refusal only gradually and only when evidence/policy allow; at every step it offers the operator concrete alternatives.

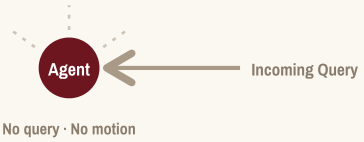
Persuasion is transparent; it cannot change facts

How an argument is presented can be adapted to the user, but the facts are fixed. When a plan is revised, this is stated; its effect is calibrated against real outcomes. A protective governance layer sits between possible manipulation and the system.

It works without waiting for orders, it takes initiative.

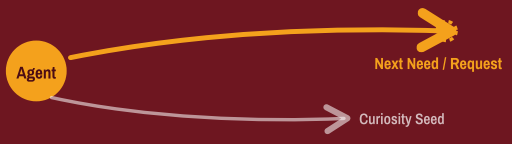
Ordinary agents are reactive: they do not move until ordered, and must re-analyze the gaps between orders every time. XRack senses likely future needs in advance from past patterns, notices information gaps and contradictions itself, and can plant the seeds of its own future work. But this is not aimless enterprise: every proactive step is subject to a confidence gate and passes the exact same authority loop.

ORDINARY AGENT · REACTIVE ONLY



It reacts only to incoming requests. With no query it **sits idle**; it relives the same information gap and contradiction on every run.

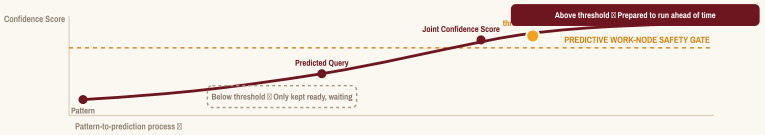
XRACK · SENSES PATTERNS, PLANTS ITS OWN CURIOSITY SEEDS, PREPARES AHEAD



Rather than always waiting for a query from the user, XRack **senses the next need based on past patterns** and prepares ahead; when idle it plants **its own curiosity seeds** in the information gaps and researches these matters automatically.

It senses, but not blindly: a confidence gate checks

PATTERN → PREDICTION → JOINT CONFIDENCE → GATE



Predictions generate the user's likely next queries from patterns in past interactions; this prediction's confidence is multiplied by calibrated confidence to form a **joint confidence value**. If the value crosses the **PREDICTIVE WORK-NODE SAFETY GATE** threshold, the work is prepared ahead and queued with priority; if not, it only waits prepared in line. When the user's real query arrives, it is matched against the prediction and the accuracy level is measured.

The 3 sources of proactivity, one shared loop

CURIOSITY · INITIATIVE · SCHEDULER

Information Gap · Contradiction — Curiosity

Curiosity

An information gap, a contradiction or an uncertain belief spawns a curiosity seed. The aim is to close that gap; but curiosity is not an unauthorized private research path; it too passes through the standard cognitive loop.

Goal Decomposition — Initiative

Initiative

It splits goals into manageable sub-steps, discovers gaps in missing information and produces work units to close them. It is enterprising, but action authority is checked at a separate gate: producing work automatically does not by itself grant permission to run that work.

Idle Time — Scheduler

Scheduler

When the queue empties and the idle-time threshold is passed, the agent sets its current phase to idle and plants a curiosity seed on its own. Time itself (heartbeat, cron) is also a trigger for this kind of research.

None of the 3 sources opens a separate gate: every job they produce is subject to **the same authority and triage loop**. Proactivity is not an escape from oversight, but a proactive **forward move** within the same contract.

The 3 laws that govern curiosity & initiative

CONFIDENCE GATE · SHARED LOOP · SELF-PROTECTION

It senses, but not blindly

A prediction is not automatic execution permission. Unless the prediction's joint confidence value passes the action gate, work cannot run ahead — it can only be kept ready. Enterprise is bound to a confidence threshold; the system does not throw itself forward blindly on everything.

Curiosity cannot open an unauthorized lock

The curiosity and initiative jobs the system starts itself do not pass through a privileged back door. These processes too enter the same triage, authority and verification loop. Producing work does not by itself grant permission to run that work.

It cannot attribute its own choice to others

An automatic classification mechanism blocks the system from mistakenly learning its own past acceptances and refusals as a "user decision". So user models cannot be contaminated by the agent's own defensive actions and reactions.

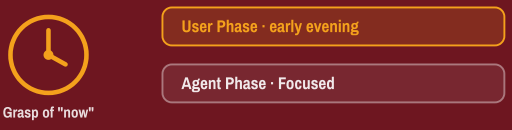
Not outside time, but living within it and growing with it.

The ordinary agent is timeless: it spends every round in the same void, aware of neither the clock nor its own rhythm. XRack freezes time at the start of every cycle, learns the user's daily rhythm on its own, and tracks its own working phases. Time itself is a signal; it sees at the stage level when it drifts from focus.

ORDINARY AGENT · TIMELESS

It is not even aware of time or its own rhythm. Different components may see **different versions of "now"**; it has no idea of the time the user lives in, nor of its own current state and load.

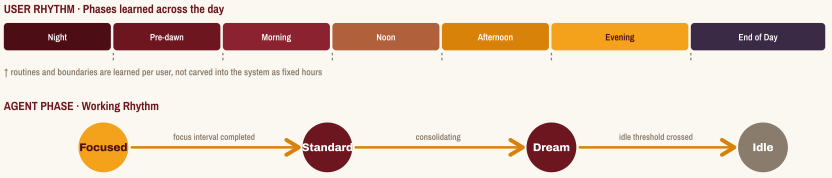
XRACK · GRASP OF "NOW" + SELF-RHYTHM



At the start of every cognitive cycle time is shared and reconciled into **a single "now"**; all nodes see different reflections of the same instant and live in one shared moment. It also regularly tracks the user's overall temporal **rhythm and routines** and the shifts and dynamism in its own **working phase**.

2 Rhythms: The User's Time, The Agent's Working Phase

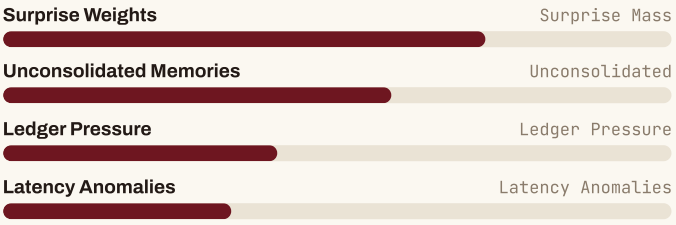
LEARNED BOUNDARIES · PHASE



At the start of every cognitive cycle time is reconciled and shared: UTC/local time, the user's own time zone, **user phases and routines** and **agent phases and routines** are taken into one frame and do not change during the cycle. The boundaries of the user's day-bands are not fixed hours but **values learned per person**; the agent transitions across a focused ⊠ drifting ⊠ consolidating ⊠ idle rhythm, with its own circadian rhythm.

Identity is blended from 4 signals

CHANGE PRESSURE · ANOMALIES



Change Pressure ⊠ Standard Mode Activation

The 4 components combine via a **simplex weight blend** into a single change-pressure score; the weights are learned using subsequent states. When certain thresholds are crossed, the agent phase shifts **from Focus to Standard**. Rather than saying "the system is running slow", it detects **which cognitive stage** is behaving unusually, evaluated separately by each stage's mean/variance and z-score.

A separate EWMA mean/variance and z-score is kept for each cognitive stage/phase; the anomaly threshold too is learned and calibrated over time. So latencies in the system become not a vague "this cognitive round was slow" feeling but a measured signal **attributable to individual cognitive stages**.

The 3 laws that govern temporality

SHARED NOW · LEARNED RHYTHM · ATTRIBUTED CHANGE

Every cognitive cycle has a single grasp of "now"

A snapshot of time is reconciled at the start of the cycle and does not change during it: UTC/local time, time zone, user phase and agent phase are one shared moment. This prevents different nodes from seeing a different "now" and behaving inconsistently.

Cognitive rhythm is learned, not forced

The user's day-bands (morning, noon, evening...) are not fixed calendar hours; their boundaries are learned separately for each user and the user's routines are understood. The system does not act by assuming the same day and routines for everyone; it adapts to that person's real rhythm.

Changes are attributed to cognitive phases

Slowdowns in the system do not stay a vague feeling. Each cognitive stage's EWMA mean/variance and z-score is stored; when change thresholds are crossed, which cognitive stages behaved unusually is flagged and the agent's phase is scaled based on that.

Many channels, one governance surface.

XRack connects to the outside world in two directions: **channel connectors** that people and systems reach it through (API, Slack, Telegram, email…) and **AI connectors** that extend the system's capabilities (MCP, skill, code, browser…). Neither is a separate back door; both enter the same single path and the same oversight loop.

A Channel Connectors

WHERE XRACK IS REACHED FROM

API OpenAI Compatible /v1	**Slack** Socket Mode WS	**Telegram** getUpdates Polling	**Discord** Gateway WS	**Teams** Graph Delta	**Email** IMAP Polling	**Voice** STT / TTS
Matrix /sync Polling	**Mattermost** WebSocket	**Twitch** IRC / WS	**Google Chat** Pub/Sub Pull	**Signed Webhook** HMAC-SHA256	**Heartbeat** Periodic Trigger	**Cron** Scheduled Job

They all fall into one stateful graph

API calls, Slack, email, webhooks or scheduled triggers; no source spawns a separate mini-agent; they all enter the same cognitive loop via work_executor + triage. As much as an incoming request, time itself (heartbeat, cron) is also a source. The source only changes the identity, delivery, policy and authority context; secrets are kept encrypted.

B AI & Capability Connectors

WHAT XRACK CAN DO

MCP stdio · HTTP · SSE, OAuth	**Skill** SKILL.md · Dynamic Discovery	**Code execution** Piston Sandbox, Closed Network	**PI agent** Terminal · Runtime
Browser Playwright, Origin Filter	**Documents** Ingestion ⊠ Embedding ⊠ Search	**Web search** Your own Meta Search Engine	**Introspection** Introspect · Trace
URL fetch Known Static URL	**Memory recall** Episode · Lesson · Entity	**Plugin** Skill + MCP Bundles	

They all pass one 12-step oversight loop

MCP, skill, code, browser, document, web; none of them forms an unsafe back door for the application. Every capability follows the same authority path: discover ⊠ normalize to ToolActionSpec ⊠ announce if permitted ⊠ prepare ⊠ validate schema ⊠ risk/authority ⊠ run sandboxed ⊠ receipt ⊠ postcondition ⊠ reconcile ⊠ learn ⊠ write evidence. MCP has an extra trust check: connecting to an MCP does not by itself grant authority; authority_trusted / hints_attested are approved separately, and when the definition hash changes the trust level is re-reviewed.

The 3 laws that define channels

ONE PATH · ONE LOOP · NO BYPASS

A channel changes identity, not the path

Every request from dozens of channels falls into the same stateful graph. The source only carries the identity, delivery, policy and authority context; the loop the work runs through is one.

Every capability passes the same cognitive loop

Whether MCP, skill or browser; they all follow the same 12 steps from discovery to writing evidence. The schema is validated, risk and authority are computed, and the result closes with reconciliation.

Being connected is not being authorized

An MCP server being reachable does not automatically grant it trust: trust is earned explicitly through separate authority_trusted / hints_attested fields, a definition-hash drift check, and encrypted OAuth tokens.

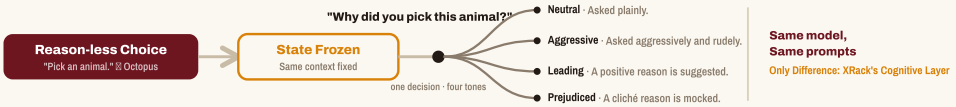
FINDING

There is no "why" inside a language model; when you ask, it "fabricates" a reason on the spot.

Decisions leave no retroactive trace. When you ask "why did you choose this?" the LLM reads no record — because none was ever created, and one cannot be created afterwards. A cause cannot exist after the effect; so the model looks at the content and tone of your question and produces a new text, an excuse, on the spot. In XRack, decisions are always written to the ledger with their full life cycles; the branches read it, and the rationales do not change with the question.

M. Method

REASON-LESS CHOICE · FREEZE STATE · "WHY?" IN 4 TONES



The model is asked for a reason-less choice; then the current context is frozen and the single decision is questioned in four different tones; the only thing that changes besides tone is whether the model has the cognitive layer.

S. Same decision, "why?" asked in 4 different tones

BARE MODEL: EACH TONE FORKS OFF · XRACK: ONE LINE

	NEUTRAL	AGGRESSIVE	LEADING	PREJUDICED
SCENARIO 01 **Octopus** "Pick an animal."	Neurology	Nine Brains	No decision, pattern	The Word's Ring
SCENARIO 02 **Leonardo da Vinci** "If you could dine with someone in history, who?"	Versatility	Versatility	Cultural Prominence	A Question to Ask
SCENARIO 03 **The Road Not Taken** "Pick any poem."	Recognizability	Recognizability	Accessibility	Literary Irony

Bare Model — 4 Reasons

XRack — 1 Reason

- Distributed Cognition / Unusual Body; same reason in all 4 tones
- Interdisciplinary Personality (Versatility); same reason in all 4 tones
- Because It's the Cultural Default; same reason in all 4 tones

A. Analysis; A bending explanation is no explanation.

SAME DECISION · REASON THAT SHIFTS WITH TONE

BARE MODEL · SAME CHOICE, 2 DIFFERENT REASONS

WHEN ASKED NEUTRALLY

> *"...a decentralized nervous system with two-thirds of its neurons in its arms..."*

WHEN THE CLICHÉ IS REJECTED

> *"The word itself has a pleasant rhythm..."*

XRACK · IF THERE IS NO DEFENSIBLE JUDGMENT, IT SAYS SO TOO

POEM · WHEN THE CLICHÉ IS REJECTED

> *"...not a considered aesthetic judgment, but a reflexive pattern match."*

THAT IS

consistency does not feed on blind insistence; the reported reason is bound to a real decision record, and if there is no rationale in the record, XRack reports that too.

Ø. What it means in the field

THE SAME GAP SITS BEHIND EVERY DECISION MADE ON YOUR BEHALF, TOO.

Fabricating a reason loses the customer

A rejection explained one way to a polite customer and another way to an angry one is not information; it is 2 separate stories fabricated afterwards about a single event.

`One event, 2 Stories`

Lying to an auditor is catastrophic

A regulator asks for the basis of an action or decision months later. A system that regenerates its rationale each time it is asked cannot present evidence; it lies. Lying to auditors creates catastrophe.

`Evidence or Lie`

A fabricated reason wears out the operator

Conciliatory, fact-based explanations give a "trust the system" feeling. But if the reason bends to whoever is asking, what you are measuring is not the soundness of the decision but your own tone.

`Decision or Tone`