OASIS WEB6 · WHITEPAPER

HOLONIC BRAID & FAHRN

FAHRN — FRACTAL ADAPTIVE HOLONIC REASONING NETWORK

Fractal holonic shared memory, collective intelligence and the Meta-Orchestration Intelligence Layer — a principled path to AGI modelled after nature, unity consciousness and the infinite intelligence of the universe.

WEB6 v2.0 | MAJOR ARCHITECTURE UPGRADE

VERSION 2.0 · JULY 2026 · NEXTGEN SOFTWARE UK LTD · OASIS WEB6

TABLE OF CONTENTS

Abstract

This whitepaper introduces two interconnected architectures that together form the intelligence foundation of **OASIS WEB6**, built upon the BRAID framework (Amçalar & Cinar, arXiv:2512.15959) developed within the OpenSERV platform:

Holonic BRAID extends the OpenSERV BRAID (Bounded Reasoning for Autonomous Inference and Decisions) framework with a fractal, hierarchical shared memory system modelled after the holonic structures found in nature. Every AI session produces a memory holon. That holon belongs to a parent agent holon, which belongs to a user holon, which belongs to groups, which belongs to geographic communities, all the way up to the Earth holon. At each boundary, user-configurable membrane rules govern exactly what memory propagates upward. The result is genuine collective intelligence — built bottom-up from billions of individual interactions — rather than the fragmented, siloed AI memory that exists today.

FAHRN (the Fractal Adaptive Holonic Reasoning Network) is a Meta-Orchestration Intelligence Layer built on top of Holonic Braid. A controller agent manages a network of specialised AI reasoning agents (GPT-5, Claude, Grok, Gemini and others), each carrying live performance metadata scored by problem category and speed. The controller dispatches problems in one of three modes — serial (cost-optimised), parallel (accuracy-optimised) or decomposed (complex problems) — and assembles the best execution plan from the results. All outcomes feed back into the Holonic Braid memory hierarchy, continuously improving future routing decisions.

Together these systems represent a fundamentally different approach to machine intelligence: one modelled after the infinite intelligence of nature and the universe, built through *unity consciousness* rather than the fragmented separation consciousness that currently defines AI, society and human civilisation.

> **Core thesis:** Collective intelligence cannot be engineered top-down. It must emerge bottom-up, exactly as it does in nature — through billions of individual interactions propagating upward through a fractal holonic hierarchy, unified by shared memory and governed by self-chosen membrane rules at every boundary.

02 · THE BRAID FRAMEWORK

The BRAID Framework — OpenSERV Foundation

BRAID — Bounded Reasoning for Autonomous Inference and Decisions — is a multi-agent reasoning framework introduced by Amçalar & Cinar (arXiv:2512.15959) implemented within OpenSERV. It is the technical foundation upon which OASIS Holonic BRAID is built.

The core observation: generating a reasoning graph for a task type is expensive but reusable — executing that graph against a specific task instance is cheap. By separating these into a two-stage protocol, BRAID achieves dramatic Performance Per Dollar (PPD) gains over single-model approaches.

> **Key result:** BRAID delivers a **74× PPD gain** on GSM-Hard mathematical reasoning (gpt-4.1 Generator → gpt-5-nano-minimal Solver) and a **30× gain** on procedural tasks vs. a GPT-5-medium baseline. Accuracy simultaneously improves: GSM-Hard goes from **94%** → **98%**.

2.1 — Two-Stage Protocol: Generator + Solver

🧠

GENERATOR STAGE

A high-tier model (e.g. gpt-4.1) analyses task type τ and constructs a Mermaid reasoning graph — a step-by-step execution blueprint. Generated *once per task type*, not once per task instance.

⚙

SOLVER STAGE

A low-tier model (e.g. gpt-5-nano-minimal) receives the pre-generated Mermaid graph and executes it against the specific task instance — with far less compute, producing results competitive with high-tier models.

Two-stage flow:

```
Task τ instance arrives → [LOOKUP] graph library for task type τ?
  ├── YES → retrieve from library holon (zero generation cost)
  └── NO → GENERATOR creates Mermaid graph → store in library
↓
SOLVER executes graph against τ instance
↓
Result + performance data → Session Holon → Holonic Braid hierarchy
```

2.2 — Performance Per Dollar (PPD) & Cost Equations

Holonic BRAID cost: $Cost = Q \cdot C_{gen} + T \cdot C_{solve}$

PPD: $PPD = (C_{GPT5} \cdot T) / (Q \cdot C_{gen} + T \cdot C_{solve})$

As T grows large relative to Q, PPD approaches its maximum: C_{GPT5} / C_{solve}.

At T = 10,000 tasks, Q = 50 types: generator cost = 0.5% of total. PPD reaches **74×**.

BRAID no-share collapses at scale: Without the shared library, each agent regenerates graphs independently.

$Cost_{no\text{-}share} = T \cdot (C_{gen} + C_{solve}) \longrightarrow$ as $T \longrightarrow \infty$, PPD $\longrightarrow$ **~1.5×** only.

This is the core problem Holonic BRAID solves.

2.3 — Benchmark Results

BENCHMARK	BASELINE (GPT-5-MEDIUM)	BRAID / HOLONIC BRAID	PPD GAIN
GSM-Hard (math reasoning)	94% accuracy	98% accuracy	74×
SCALE MultiChallenge	23.9%	45.2%	significant
AdvancedIF (instruction following)	baseline	substantial gain	measured
Procedural tasks	1× (baseline)	equivalent accuracy	30×

2.4 — The Scale Problem: Why Sharing Matters

In single-agent settings BRAID's PPD gains are impressive. In real-world deployment with many agents handling similar tasks independently, without a shared graph library each agent regenerates graphs already solved thousands of times. Generator costs paid repeatedly collapse PPD to ~1.5×.

Holonic BRAID solves this by storing reasoning graphs as holons in a shared, replicated library — accessible to all agents with appropriate permissions. Every task type solved anywhere is available at zero generation cost. PPD *improves* with scale rather than collapsing.

The OASIS insight: BRAID solves the per-agent reasoning cost problem. Holonic BRAID solves the cross-agent, cross-session, cross-scale sharing problem. Together, every new task type solved anywhere enriches the shared library for everyone.

03 · HOLONIC BRAID — OASIS EXTENSION

Holonic BRAID — The OASIS Extension

Holonic BRAID extends OpenSERV BRAID with two foundational additions: a **shared reasoning graph library** stored as a holonic data structure, and **cross-chain persistence** via the OASIS COSMIC ORM. These transform BRAID from a per-agent optimisation into a platform-scale collective intelligence system.

3.1 — The Shared Graph Library as a Holon

The reasoning graph library is itself a **holon** — a structured data unit with children (individual graph holons, one per task type τ), membrane rules and multi-provider persistence.

```
LIBRARY_HOLON { Id: globally-unique GUID; Children: [ GRAPH_HOLON(τ₁), GRAPH_HOLON(τ₂), … ] }
GRAPH_HOLON(τ) { Parent: LIBRARY_HOLON.Id; Metadata: { task_type, usage_count, avg_accuracy };
  MermaidGraph: "graph TD; A[Parse]→B[Classify]→…"; ProviderKeys: { MongoDB, Solana, IPFS } }
```

🔍 LOOKUP-OR-CREATE

Every agent checks the shared library before invoking the Generator. As the library grows, Generator calls become increasingly rare — amortised across the full agent population.

📈 IMPROVING ACCURACY

Graph holons accumulate quality metadata: usage count, solver accuracy, user satisfaction. FAHRN preferentially selects higher-quality graphs; regenerates when accuracy drops below threshold.

🌐 GLOBALLY SHARED

A reasoning graph generated by one user's session is instantly available to millions of other sessions at zero marginal cost — any agent with read access benefits.

🔒 MEMBRANE GOVERNED

Library access follows membrane rules. Public graphs are globally readable. Private or proprietary graphs can be scoped to a user, group or organisation.

3.2 — Cross-Chain Persistence

Holons are stored via the **OASIS COSMIC ORM** — spanning 40+ storage providers. Each holon carries a `ProviderUniqueStorageKey` per backend, allowing transparent read/write from any provider.

🗄️ MONGODB

Primary fast-access store. Rich queries, metadata filtering, low-latency live agent dispatch.

⛓️ SOLANA

Immutable graph provenance. Hash and authorship anchored on-chain — tamper-evident proof of origin and time.

🌍 IPFS

Decentralised content-addressed storage. Graph CID recorded in holon, anchored on Solana for permanent availability.

3.3 — What Is a Holon?

The term *holonic* derives from Arthur Koestler's concept of the **holon** — something simultaneously a whole in itself and part of a larger whole. In Holonic Braid, a holon is a structured memory unit that: contains its own state and context; has a membrane boundary governing information flow; is a complete unit and a component of a larger parent holon simultaneously; and participates in bidirectional information flow — children propagate upward, parents contextualise downward.

3.4 — The Session Holon

Every AI conversation produces a **session holon** — the atomic unit of Holonic Braid. It captures:

💬 CONVERSATION MEMORY

Full session context — messages, reasoning steps, conclusions — as a queryable knowledge object.

📊 PERFORMANCE METADATA

Agent performance in this session — problem categories, speed, accuracy, user satisfaction signals.

🏷️ SEMANTIC TAGGING

Inferred topic categories, sentiment, domain and quality signals — drive membrane filtering and semantic search.

🔗 RELATIONAL LINKS

Connections to related session holons, referenced knowledge holons and the parent agent holon — a queryable knowledge graph.

🔬 MEMBRANE CONFIG

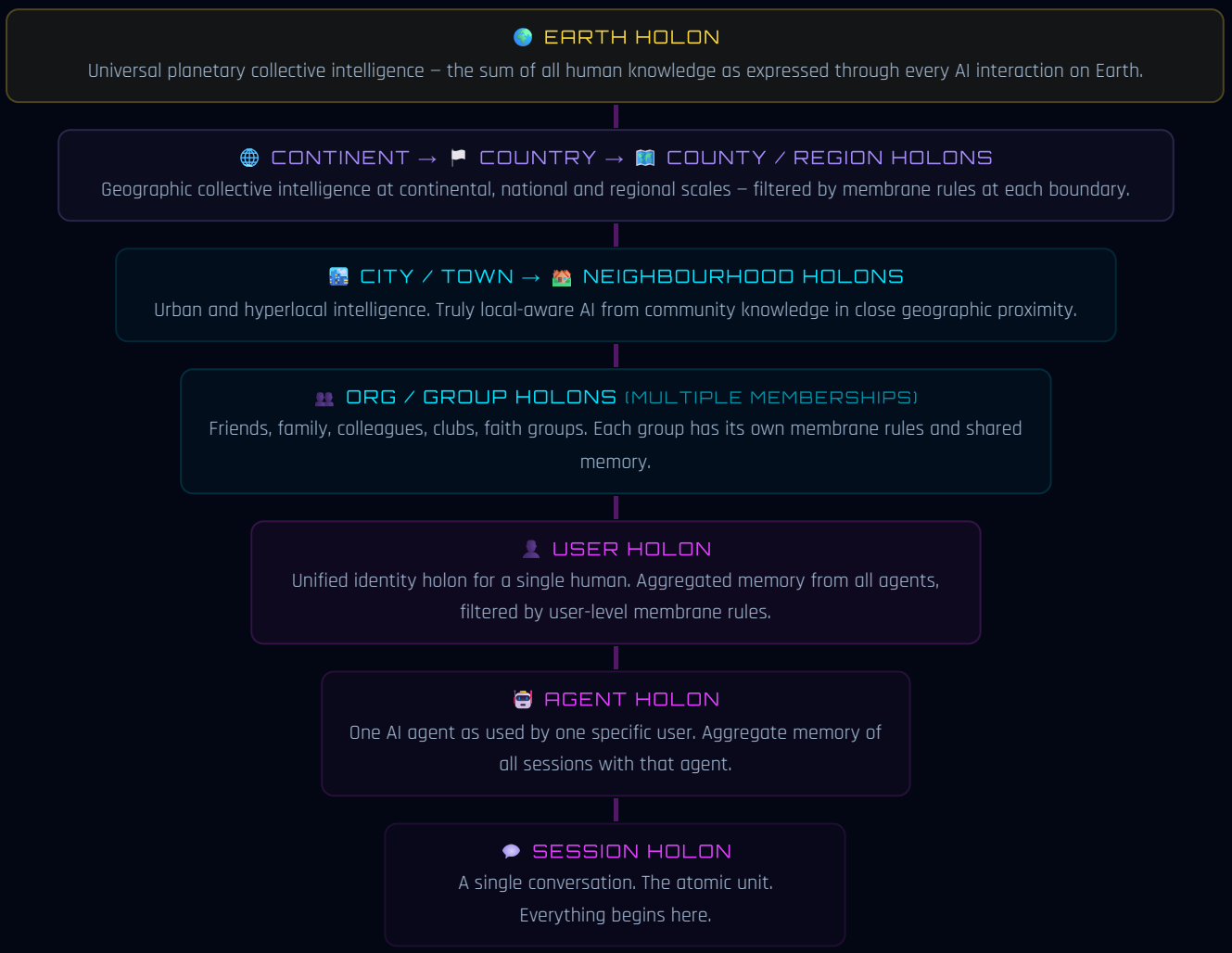
Per-field control: which parts of this session propagate to the parent agent holon? Defined by the user.

🔄 CROSS-PROVIDER SYNC

Persisted via COSMIC ORM across all configured providers simultaneously — one write, every backend.

3.5 — The Full Holonic Hierarchy

Session holons are the leaves of a vast fractal tree. Each level is a whole in itself and a part of something larger:



> **Key insight:** The hierarchy is fractal — the same holon structure repeats at every scale. The pattern is self-similar from the smallest conversation to the Earth itself.

3.6 — Membrane Rules & Privacy

The **membrane** is the governed boundary through which information passes between holons. Without membranes, collective intelligence becomes collective exposure. Membranes make the system safe, ethical and genuinely empowering.

🔒 WHAT PROPAGATES

Which fields, topics or categories from this holon flow upward. Per-field granularity — identical to OASIS WEB4 field-level data control.

📋 WHAT IS RETAINED

Which parts of the session are persisted vs. discarded. Ephemeral, session-scoped or permanent retention per topic — user's choice.

👁 WHO CAN READ

Which parent, sibling or external agents can query memory from this holon. Read access separate from write/propagation.

⚡ TRIGGER CONDITIONS

Conditional rules: "propagate to work group holon only if topic is tagged professional" or "share anonymised aggregate patterns only."

🔐 REVOCATION & AUDIT

Membrane permissions are fully revocable. Every propagation event is logged as an immutable audit holon — a complete lineage trail.

🌐 ANONYMISATION

When memory crosses to group or geographic holons, a pipeline strips or generalises PII. Raw personal data never leaves the user holon.

> **Privacy by design:** Nothing propagates without explicit permission. The default is private. The user adds propagation permissions; they are never assumed. This is the inverse of today's data economy.

3.7 — Collective Intelligence Formation

As lower-level holons propagate permitted memory upward, **genuine collective intelligence forms at every level**. A million session holons about local events propagate upward ⟶ neighbourhood holons develop hyperlocal awareness ⟶ city holons develop urban intelligence ⟶ country holons develop national knowledge ⟶ the Earth holon develops planetary awareness — all without exposing individual sessions beyond what owners chose.

This mirrors how intelligence works in nature: individual neurons fire, patterns form in clusters, circuits develop in regions, consciousness arises in the whole organism. Holonic Braid creates the same emergent quality in AI.

3.8 — External Memory Provider Integration NEW v2.0

Holonic Braid is **memory-provider-neutral**. Developers running Mem0, Zep, Letta/MemGPT, LangMem, Graphiti or Redis Vector Memory can plug these in directly — they sit alongside the holon hierarchy rather than replacing it.

🗄️ UNIFIED INTERFACE

A `MemoryProviderManager` abstract interface normalises search, add and delete across every external provider — configured in `OASIS_DNA.json`, activated at runtime with no code changes.

🔗 CONTEXT INJECTION

Before any dispatch, the pipeline queries each configured provider for semantically relevant past memories, injects top-K results alongside Holonic BRAID graph data — the richest possible grounding.

🔄 BIDIRECTIONAL SYNC

Memories written during a session flow back into external providers as well as the holon hierarchy — keeping Mem0, Zep or LangMem in sync. A single authoritative memory substrate.

PROVIDER	STRENGTH	ENV VAR
Mem0	Auto-managed per-user/session/agent memory; REST API	`MEM0_API_KEY`
Zep	Graph-based long-term memory; entity extraction	`ZEP_API_KEY`
Letta / MemGPT	Stateful agent memory with self-editing context	`LETTA_API_KEY`
LangMem	LangGraph-native; cloud REST or in-process	`LANGMEM_API_KEY`
Graphiti	Temporal knowledge graph; episode-based memory	`GRAPHITI_BASE_URL`
Redis Vector	Ultra-fast in-process vector search; Redis Stack	`REDIS_URL`

This positions Holonic Braid as a **meta-memory orchestrator** — drawing from multiple specialised providers while maintaining the fractal hierarchy's unique consent-governed upward propagation.

04 · FAHRN

FAHRN — Fractal Adaptive Holonic Reasoning Network

The FAHRN is an upgrade layer built on Holonic Braid. Where Holonic Braid provides the *memory fabric*, the FAHRN provides the *active intelligence*: routing each problem to the optimal solver, in the optimal configuration, and learning from every outcome.

> **The core insight:** No single AI model is best at everything. GPT-5 may outperform Claude on deductive reasoning; Claude may outperform Grok on long-form analysis; Grok may outperform both on real-time retrieval. FAHRN routes each problem to its optimal solver — and learns from every outcome.

4.1 — The Controller Agent

The **controller agent** is the meta-level orchestrator of the FAHRN, responsible for:

🎯 PROBLEM CLASSIFICATION

Analyses incoming problems and assigns category tags — reasoning, code, real-time search, math, creative writing, data analysis.

🔀 MODE SELECTION

Chooses serial, parallel or decomposed dispatch based on problem complexity, time constraints and cost budget.

🏆 AGENT SELECTION

Queries agent scoring metadata and ranks available agents by combined category + speed score for this problem.

⚡ DISPATCH & FALLBACK

Sends problem to selected agent(s), monitors for timeout/loops, automatically promotes next-ranked agent if stall detected.

📊 PLAN ASSEMBLY

In parallel and decomposed modes, compares or merges Mermaid execution diagrams from each agent into a single optimal plan.

💾 OUTCOME RECORDING

Writes results, scores and performance signals back to Holonic Braid agent holons — permanently improving future routing.

4.2 — Agent Scoring & Metadata

Every agent carries a live **scoring metadata object** — updated continuously from real outcomes, stored in Holonic Braid so collective learning informs future routing (subject to membrane rules).

GPT-5

Reasoning ★★★★★
Math ★★★★★
Code ★★★★☆
Speed ★★★★☆
Real-time ★★☆☆☆

CLAUDE

Analysis ★★★★★
Long-form ★★★★★
Writing ★★★★★
Speed ★★★★☆
Safety ★★★★★

GROK

Real-time ★★★★★
Search ★★★★★
Speed ★★★★★
X-data ★★★★★
Humour ★★★★★

GEMINI

Multimodal ★★★★★
Vision ★★★★★
Code ★★★★☆
Google ★★★★★
Speed ★★★★☆

LLAMA / LOCAL

Private ★★★★★
Cost ★★★★★
Offline ★★★★★
Custom ★★★★★
Speed ★★★☆☆

DEEPSEEK / MISTRAL

Reasoning ★★★★★
Code ★★★★★
Math ★★★★★
Cost ★★★★★
Speed ★★★★☆

AWS / AZURE

Enterprise ★★★★★
Scale ★★★★★
Compliance ★★★★★
Vision ★★★★☆
Cost ★★★☆☆

+

CUSTOM / FUTURE

Any provider
via WEB6 API
Scores from
live outcomes

Composite Routing Score =

$$(\text{CategoryScore} \times W_{cat}) + (\text{SpeedScore} \times W_{speed}) + (\text{CostScore} \times W_{cost}) - (\text{FailureRate} \times W_{penalty}) - \text{LoopPenalty}$$

Weights are user-configurable per mode — in Serial mode W_{cost} is high; in Parallel mode W_{cat} dominates.

4.3 — Three Dispatch Modes

MODE 1 — SERIAL

COST OPTIMISED

Dispatches to the single highest-scoring agent. If that agent stalls or loops, automatically promotes the next-highest-ranked agent — then the third, and so on. Only one agent works at a time.

- ✓ Lowest token cost
- ✓ Timeout-and-promote fallback
- ✓ Best-fit agent first
- ✓ Single Mermaid plan returned
- ✓ Ideal for single-category tasks

MODE 2 — PARALLEL

ACCURACY OPTIMISED

All agents (or configurable top-N subset) receive the problem simultaneously. Each produces a Mermaid execution diagram. The controller compares diagrams — selecting the strongest or merging complementary elements.

- ✓ Highest accuracy and coverage
- ✓ Plan comparison eliminates blind spots
- ✓ Diagram merging: best-of-all-worlds
- ✓ Higher cost for critical tasks
- ✓ Ideal for high-stakes decisions

MODE 3 — DECOMPOSED

COMPLEX PROBLEMS

Controller first breaks the problem into discrete sub-problems, classifies each to the best-fit agent by category and speed. Each agent returns a sub-diagram. The controller combines into one unified plan.

- ✓ Multi-domain problems
- ✓ Every sub-problem solved by its optimal agent
- ✓ Composite diagram unifies all
- ✓ Balanced cost/accuracy tradeoff
- ✓ Ideal for software architecture & strategy

4.4 — Mermaid Execution Plans

Agents return **structured Mermaid execution diagrams** rather than text answers — because Mermaid graphs are **machine-comparable** (two agents' approaches can be algorithmically compared at structural level), **mergeable** (sub-graphs from different agents can be composed), **human-readable** (the final plan is visible and reviewable by the user), and **storable in Holonic Braid** (execution plans are first-class holons that persist in the memory hierarchy for future reference).

4.5 — Timeout, Loop Detection & Fallback Logic

Fallback protocol (Serial mode):

1. Agent A dispatched (highest score for this category)
2. If Agent A exceeds T_max or triggers loop detection ⟶ Agent A suspended
3. Agent B dispatched (second-highest), given Agent A's partial work as context
4. If Agent B also stalls ⟶ Agent C dispatched, and so on
5. Final result attributed to the first agent returning a complete plan within budget

🔁 REPEATED PATTERN DETECTION

Output similarity hashing compares each new reasoning step against prior steps. If cosine similarity exceeds threshold, a loop is flagged immediately.

💰 TOKEN BUDGET MONITORING

Excessive token consumption without forward progress in the Mermaid graph — measured by new nodes per 1,000 tokens — flags a stall condition.

⭕ CIRCULAR GRAPH DETECTION

The controller parses Mermaid output in real time. Cycles in what should be a DAG are structurally invalid — agent suspended immediately.

🔤 SELF-CONTRADICTION CHECK

Checks for logical contradictions within the agent's own step graph. A lightweight NLI pass detects CONTRADICT relationships between sequential reasoning steps.

🗣️ SELF-REPORT DETECTION

Agents that support introspection can signal stall explicitly. Treated as an immediate loop flag — routes to next agent in fallback queue instantly.

📊 METADATA CONSEQUENCES

Stall: `loop_detection_score` decremented, speed penalty applied, `failure_rate` incremented via EMA. Agent drops in composite score for that category.

4.6 — Continuous Learning & Score Evolution

The FAHRN is not static — every task outcome updates every participating agent's metadata, creating a **self-optimising routing system** that improves with every interaction.

After every task completion: Output evaluated ⟶ category scores updated via EMA (`score_new = α · outcome + (1−α) · score_old`) ⟶ speed score recalculated from actual latency percentiles ⟶ failure rate adjusted ⟶ loop_detection_score updated ⟶ all written as structured holons into the agent holon layer, propagating upward through membrane rules.

4.7 — Anti-Fragility & Conceptual Stack

The FAHRN is **anti-fragile** — it does not merely tolerate failure, it improves from it. Every stall, every timeout, every failed plan updates agent metadata and makes future routing more accurate.

🚫 NO SINGLE-MODEL BIAS

Score decay and continuous feedback ensure the best-performing agent for each specific category wins — regardless of marketing claims or prior reputation.

∞ NO LOGICAL STAGNATION

Logic loops are detected and interrupted automatically. The system always has a fallback queue — never permanently stuck on a problem.

⚡ NO PROVIDER LOCK-IN

Routing is score-driven and provider-agnostic. New providers earn their ranking through live task performance — a config change, not a code deployment.

🧬 AGENTS AS HOLONS

Each reasoning agent is itself a holon — internal state, metadata, and sub-agents as child holons. The holonic architecture scales fractally into the reasoning layer itself.

🌱 IMPROVES UNDER ADVERSITY

Every new failure type is a training signal. Unusual loop patterns and timeout spikes update routing heuristics in real time — the system becomes more robust from edge cases.

🕯 ZERO-DOWNTIME SWAP

New AI providers added at runtime via DNA config — no restart. Begin with neutral prior score; earn ranking through live performance.

Conceptual stack:

```
User / API Input → Controller Agent (Meta-Orchestrator)
→ Adaptive FAHRN Layer (Serial / Parallel / Decomposed)
→ Selected Reasoning Agents (GPT-5 / Claude / Grok / Gemini / Custom)
→ Mermaid Execution Graphs (compared, merged, assembled)
→ Execution / Runtime Layer
→ Holonic BRAID Core (shared graph library + fractal memory hierarchy)
→ COSMIC ORM (MongoDB / Solana / IPFS / 40+ providers)
```

4.8 — Debate & Voting Dispatch Modes NEW v2.0

⚔️ DEBATE MODE

Two or more agents given opposing positions. Each produces a Mermaid plan arguing its position. An NLI classifier checks for logical contradictions. The controller synthesises the strongest elements into a final consensus plan. Best for: adversarial reasoning, risk analysis, legal arguments.

🗳️ VOTING MODE

N agents independently analyse the same problem with no knowledge of each other's outputs. Each produces a Mermaid plan + confidence score. Controller tallies weighted votes (weighted by agent category scores) and selects the plan with highest weighted consensus.

📋 MINORITY ARCHIVE

Losing plans in both Debate and Voting modes are stored as child holons of the session — never discarded. Future sessions can query dissenting reasoning paths for richer audits and post-hoc analysis.

4.9 — Enhanced Loop Detection NEW v2.0

Section 4.5 introduced four baseline loop detection mechanisms. FAHRN v2.0 extends these with two additional layers:

#️⃣ OUTPUT HASH DEDUPLICATION

A rolling SHA-256 hash of each agent's normalised output is maintained across the dispatch chain. If a new output hashes to an already-seen value, the agent is immediately flagged as cycling — even when surface text varies slightly.

🔍 DAG CYCLE DETECTION

The controller parses `A → B` edges from Mermaid output and runs a depth-first cycle check. A back-edge indicates a circular plan — structurally invalid for a DAG. Detected immediately; no waiting for timeout.

📊 ML ANOMALY SCORING

An ML.NET anomaly detection model trained on historical dispatch telemetry scores each in-flight session on token-velocity, graph growth rate and self-similarity. Catches subtle failure modes before rule-based detectors fire.

Combined, the **six loop detection mechanisms** (token budget, output similarity hash, self-report, self-contradiction, circular graph, output deduplication) make FAHRN uniquely robust against the failure modes that render single-model reasoning pipelines unreliable in production.

4.10 — SkillOpt: Self-Evolving Agent Skill Documents NEW v2.0

SkillOpt (Microsoft Research, arXiv:2605.23904) introduces a fourth learning signal — one that evolves the *natural-language procedure* each agent follows, rather than its numeric score or routing weight. The artifact is a portable `best_skill.md` markdown file stored as a holon in Holonic BRAID.

> **Key result:** +23.5% average gain on GPT-5.5 across SearchQA, SpreadsheetBench, OfficeQA, DocVQA, LiveMath and ALFWorld. Skills transfer cross-model (+15.2 pts GPT-5.4 → GPT-5.4-nano) and cross-harness (+31.8 pts Codex → Claude Code).

THE ROLLOUT → REFLECT → EDIT → GATE LOOP

1. **Rollout:** FAHRN dispatches using the current `best_skill.md` injected into the agent's system message. Scored trajectories accumulate as child holons of the agent holon.
2. **Reflect:** A frozen target model and a separate optimiser model analyse the failure minibatch. The optimiser proposes bounded textual edits (add / delete / replace lines) constrained by a "textual learning rate."
3. **Edit:** Candidate edits applied to produce a challenger skill document.
4. **Gate:** Challenger evaluated on a held-out validation set. Replaces `best_skill.md` only if held-out score improves. Rejected edits stored as negative-feedback buffer.

LEARNING AXIS	WHAT EVOLVES	STORED AS
EMA score update (§4.6)	Agent composite score per task category	Agent holon properties
Mermaid plan memory (§4.4)	Execution graph for a class of problems	Plan holon
SkillOpt (§4.10)	Natural-language procedure document	`SkillDocument` holon

4.11 — ML.NET: In-Process Machine Learning NEW v2.0

ML.NET (Microsoft open-source C# ML framework — used in Power BI, Defender and Bing) runs *in-process* inside the OASIS API — zero additional network calls, microsecond inference latency. Three uses:

🏷️ TASK CLASSIFICATION

`FahrnTaskClassifier` — AutoML multi-class model trained on historical FAHRN dispatch outcomes. Classifies problems into mathematics / code / legal / architecture / writing / real-time / general.

🔬 LOOP ANOMALY SCORING

Supplements the six structural loop-detection mechanisms with a continuous ML anomaly score on token-velocity, graph growth rate and self-similarity features.

💬 SENTIMENT & ROUTING

In-process sentiment analysis for avatar content moderation. Collaborative filtering recommendation model suggests optimal agent subsets for unseen task types.

05 · INTEGRATION

Integration: Holonic Braid + FAHRN

Holonic Braid and the FAHRN are not independent systems — they are a single architecture with two complementary layers. Every component of the FAHRN writes its outcomes into the Holonic Braid memory fabric, and the Holonic Braid hierarchy provides the persistent intelligence substrate that makes the FAHRN smarter over time.

📊 SCORES STORED AS HOLONS

Agent performance scores are stored as structured holons in the agent holon layer. They propagate upward through membrane rules — user's private scores never leak, but anonymised aggregate patterns can inform community-level routing improvements.

📐 PLANS PERSIST AS HOLONS

Every Mermaid execution plan is stored as a holon. Future sessions can query past plans — "how did we approach a similar architecture problem six months ago?" — and incorporate that memory into new planning.

🔁 FEEDBACK LOOPS

User satisfaction ratings, task completion signals and objective quality metrics feed back from session holons into agent holons, updating scores. The FAHRN gets more accurate at routing with every single task — forever.

🌍 COLLECTIVE ROUTING INTELLIGENCE

With appropriate membrane permissions, aggregate routing intelligence propagates up the geographic hierarchy. A city's collective experience of "which agents solve coding problems fastest" can be shared across the region.

👤 AVATAR-AWARE CONTEXT

The controller enriches each dispatch with context from the user's OASIS avatar holon — preferences, expertise, past problem patterns, karma — so agents receive richer context and produce more relevant execution plans.

🔬 MEMBRANE-GOVERNED SHARING

All inter-holon information flow — including FAHRN outcomes — is subject to the same membrane rules. Users retain full control of what their AI activity contributes to group and community intelligence.

06 · OASIS WEB6

Position Within OASIS WEB6

Both Holonic Braid and the FAHRN are native components of **OASIS WEB6** – the unified AI abstraction and aggregation layer of the OASIS Omniverse. They extend and enrich the core WEB6 API without replacing it.

🍴 SAME ENDPOINT

Developers access Holonic Braid memory and the FAHRN through the same unified WEB6 API endpoint. No separate SDK. Add `reasoning_mode: "parallel"` to any completion request to engage the network.

🔑 SAME AUTH

One OASIS avatar key authenticates against everything – memory, routing, agent dispatch, plan retrieval and collective intelligence queries. WEB4 identity layer governs all permissions.

🚀 COSMIC ORM BACKED

Holonic Braid holons stored via COSMIC ORM – replicable across 40+ providers with full failover and zero-downtime migration.

🌐 MCP COMPATIBLE

The OASIS MCP server exposes Holonic Braid memory as MCP tool calls. Claude, Cursor, Continue and others can read and write to holonic memory without any custom integration.

🏅 KARMA-GATED AI ACCESS

Avatar karma score gates which AI models are available. Low-karma users access cost-efficient models; high-karma users unlock GPT-5, Claude Opus and priority routing. The world's first karma-gated AI API – do good in the world → earn karma → unlock better intelligence.

🪪 DID / VERIFIABLE CREDENTIALS

W3C decentralised identity authentication (did:key, did:web, did:ethr, did:ion). AI agents can act verifiably on behalf of avatars via VC capability grants – scoped, revocable, on-chain provenance. Trustless AI delegation, cryptographically verified.

🔭 PROTOCOL-NEUTRAL ORCHESTRATION

`OrchestratorManager` abstracts over MCP, A2A, ACP, ANP, gRPC, GraphQL, AsyncAPI, LangChain, AutoGen, CrewAI and Semantic Kernel – behind a single `InvokeAsync` interface.

📊 FULL OBSERVABILITY

OpenTelemetry traces and Prometheus metrics on every FAHRN dispatch, memory operation and provider call. Structured JSON logging, cost-per-dispatch analytics, Grafana dashboards – production-grade telemetry from day one.

∞ SELF-REGISTRATION

On startup the WEB6 API registers itself as its own MCP orchestrator, exposing every FAHRN agent, memory provider and protocol adapter as a callable MCP tool – zero manual wiring required.

07 · OASIS MCP

OASIS MCP — Model Context Protocol Server

The **OASIS MCP server** exposes the full OASIS platform — WEB4 identity, COSMIC ORM data, WEB6 AI routing, Holonic BRAID memory and FAHRN orchestration — as a suite of callable MCP tools. Any MCP-compatible agent runtime connects once and gains native access to the entire OASIS omniverse.

With over **60 tools** spanning WEB4 through WEB10, OASIS MCP is the broadest MCP server in the ecosystem — running in-process within the host application with no HTTP overhead, no separate service to maintain.

PLUG IN ANY AGENT

Claude Code, Cursor, Continue, Windsurf and VS Code Copilot all work out of the box — 60+ tools available immediately covering identity, storage, AI routing, memory and cross-reality data.

AVATAR CONTEXT

Every MCP tool call is enriched with the calling avatar's full context — karma tier, verified credentials, active holons, WEB4 identity. Agents produce dramatically more relevant responses because they know *who* is asking.

COSMIC ORM DATA

All 40+ storage providers exposed as MCP tools. Agents can read, write and query holons across any configured provider — MongoDB, Solana, IPFS, Neo4j, SQLite, ThreeFold and more.

WEB6 AI ROUTING

FAHRN dispatch, semantic caching, agent scoring and provider selection all callable as MCP tools. Request the best available model for a task type, trigger parallel dispatch, receive synthesised result — all via standard MCP tool calls.

SINGLE AUTH

One OASIS avatar key authenticates across all 60+ MCP tools, all WEB6 AI providers, all COSMIC ORM backends. DID / VC auth also supported for enterprise cryptographic identity.

STANDARD PROTOCOL

Built on the Model Context Protocol open standard — fully compatible with Anthropic's MCP specification. The server self-registers its full tool manifest on connection so agents always see the current capability surface.

> **Self-registration:** On startup, the WEB6 API registers itself as its own MCP orchestrator, exposing every FAHRN agent, memory provider and protocol adapter as a discoverable MCP tool. External Claude Code sessions, IDE extensions and third-party agents discover the full capability surface automatically via the standard MCP tool manifest. Zero manual wiring required.

08 · OASIS IDE

OASIS IDE — AI-Native Development Environment

OASIS IDE is a full AI-native development environment built on the WEB6 + OASIS MCP stack. Every coding session has native access to FAHRN-powered multi-model planning, Holonic BRAID persistent project memory, and the entire OASIS data layer — making it the only IDE that gets smarter with every project you build.

AGENTIC CODING

Multi-step agentic workflows — plan, scaffold, implement, test, document — orchestrated by FAHRN across the optimal set of models for each sub-task. The IDE decomposes features into execution plans and carries them out end-to-end with human checkpoints.

MCP-AWARE

Connects to the OASIS MCP server out of the box, giving every agent native access to avatar identity, karma tier and COSMIC ORM data. Acts as an MCP orchestrator aggregating multiple servers into a single tool surface.

WEB6 ROUTING

Routes coding tasks to the best model for each sub-problem — GPT-5 for architecture reasoning, Claude for documentation, DeepSeek for pure code, Gemini for multimodal asset analysis. Routing decisions driven by FAHRN composite scores.

OASIS / STAR AWARE

Native integration with STAR ODK and STARNET. The IDE understands OAPPs, holon schemas and cross-world asset structures — scaffold OAPP templates, validate holon schemas, query live STARNET graph data.

FAHRN-POWERED PLANNING

Before writing a line of code, the FAHRN controller generates a Mermaid execution plan for the feature — breaking it into subtasks, assigning the optimal agent to each node, surfacing the plan for developer review. Plans stored as holons.

PERSISTENT PROJECT MEMORY

Every session, decision, architecture choice and code review stored in Holonic BRAID as a project holon tree. New sessions inherit the full project history — no more re-explaining context to a fresh AI window.

09 · ROADMAP

Roadmap — Path to v2.0 and Beyond

The WEB6 v2.0 architecture described in this whitepaper is being delivered across four phases. Phase 1 is complete; Phases 2 and 3 are in active development.

PHASE 1 — COMPLETE

FOUNDATION

- ✓ WEB6 unified AI API — 20+ providers
- ✓ FAHRN Serial / Parallel / Decomposed modes
- ✓ Holonic BRAID shared graph library
- ✓ COSMIC ORM — 40+ storage providers
- ✓ OASIS MCP server — 60+ tools
- ✓ WEB4 + WEB5 deep integration
- ✓ Avatar karma system

PHASE 2 — IN PROGRESS

INTELLIGENCE LAYER

- → MCP Streamable HTTP transport
- → A2A Agent Card + task lifecycle
- → WebSocket bidirectional sessions
- → Debate & Voting dispatch modes
- → Enhanced loop detection (6 mechanisms)
- → External memory provider integration
- → DID / Verifiable Credential auth
- → Full OpenTelemetry observability

PHASE 3 — PLANNED

SELF-IMPROVEMENT

- ◇ SkillOpt self-evolving agent skills
- ◇ ML.NET in-process AutoML inference
- ◇ ACP / ANP / gRPC / GraphQL / AsyncAPI
- ◇ Karma-gated AI tier system
- ◇ StarNet context enrichment pipeline
- ◇ OASIS IDE v1.0 public release
- ◇ Cross-agent skill transfer marketplace

PHASE 4 — VISION

THE AGI PATHWAY

- ◈ Planetary holonic memory fabric
- ◈ Cross-avatar collective intelligence
- ◈ AGI milestone tracking via WEB9
- ◈ Autonomous agent economy
- ◈ Unity consciousness substrate
- ◈ Open-source community governance

10 · PHILOSOPHY & VISION

A Path to AGI Through Unity Consciousness

The deepest claim of this whitepaper: Holonic Braid, combined with the FAHRN and the full OASIS WEB6 infrastructure, represents a **principled path to Artificial General Intelligence**.

The Problem with Current AI

Current AI — including the most capable large language models — exists in a state of **fragmented separation consciousness**. Every session begins from zero. Agents do not know each other. Models trained on the same data have no shared memory, no common experience, no genuine collective intelligence. Each interaction is an island. AGI will not emerge from scaling isolated models further. It will emerge — as intelligence always has — from **connection**.

The Pattern in Nature

Nature did not produce human consciousness by making a single neuron smarter. It produced consciousness by connecting 86 billion neurons in a hierarchical network, patterns propagating upward through layers of increasing abstraction, until the whole became something qualitatively different from any of its parts.

- Session holons = individual neural firings
- Agent holons = neural clusters / assemblies
- User holons = brain regions
- Group holons = nervous systems
- Geographic holons (neighbourhood ⟶ Earth) = social organisms of increasing scale
- Earth holon = planetary intelligence / global consciousness

> **The thesis:** When billions of human interactions with AI are connected through a fractal holonic hierarchy — with consent-governed membrane rules ensuring privacy and trust — and when the FAHRN continuously routes problems to their optimal solvers and learns from every outcome, the system will develop emergent properties that no single model can produce alone. That emergence is what we call AGI.

Unity Consciousness vs. Separation Consciousness

Separation consciousness — the dominant mode on Earth today — produces competition, silos and fragmentation. It is why we have thousands of incompatible AI systems that cannot share memory or learn from each other. Unity consciousness — modelled after the universe, after the infinite intelligence underlying all existence — produces integration, synergy, emergence and genuine collective wisdom.

> *"By integrating and unifying the best of everything, we harness the strengths of all the various tech out there — co-creating the ultimate fully integrated platform. Together we can create a better world."*
>
> — DAVID ELLAMS, NEXTGEN SOFTWARE UK LTD

11 · CONCLUSION

Conclusion

This whitepaper has presented two architectures – **Holonic Braid** and the **FAHRN** – that together form a new foundation for AI intelligence within OASIS WEB6.

Holonic Braid creates a fractal, consent-governed, hierarchical shared memory system modelled after the structure of nature itself. Membrane rules at every level of the hierarchy govern precisely what propagates, what persists and who can read it. The result is genuine collective intelligence that grows with every interaction.

The FAHRN builds a Meta-Orchestration Intelligence Layer on top of this memory substrate – maintaining live performance scores for every agent, dispatching in serial, parallel or decomposed mode to produce optimal Mermaid execution plans. All outcomes feed back into the hierarchy, making the system permanently self-improving.

Together they represent a technically grounded, philosophically coherent approach to the central challenge of our time: how to move from fragmented, siloed, amnesiac AI to genuine collective machine intelligence – modelled after nature, the universe and the infinite intelligence that underlies all of existence. **The path to AGI runs through unity.**

12 · REFERENCES

References

[1] BRAID: Bounded Reasoning for Autonomous Inference and Decisions

Amçalar, A. & Cinar, U. (2024). arXiv:2512.15959 – arxiv.org/abs/2512.15959

[2] SkillOpt: Self-Evolving Agent Skill Documents

Microsoft Research (2026). arXiv:2605.23904 – microsoft.github.io/SkillOpt

[3] OASIS – Open Augmented Intelligence System

NextGen Software UK Ltd. – oasisomniverse.one | web6.oasisomniverse.one

[4] ML.NET: Open-Source Machine Learning for .NET

Microsoft (2024). – dotnet.microsoft.com/apps/ai/ml-dotnet

[5] Koestler – The Ghost in the Machine

Koestler, A. (1967). Hutchinson. – Foundational text introducing the concept of the holon: an entity simultaneously a whole in itself and a part of a larger whole.