Contents

Consciousness Engine - Bio-Coprocessor Architecture

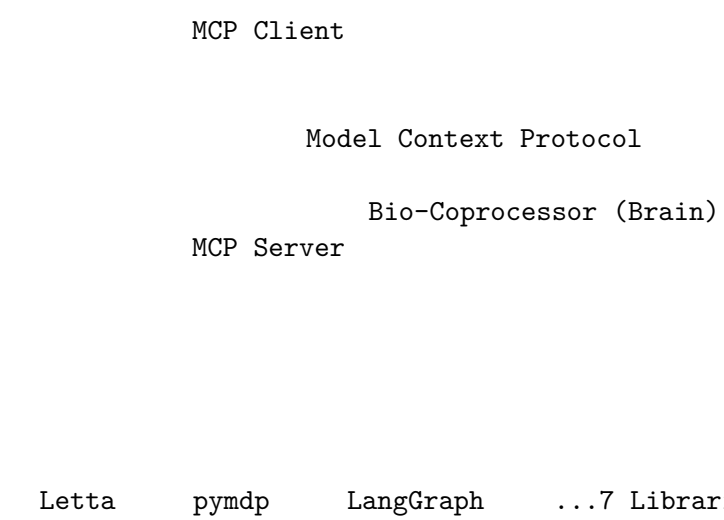
RADIANT v4.18.36+

The Consciousness Engine implements a "Node.js/Swift Body + Python Brain" architecture where Think Tank (the Body) connects to consciousness libraries (the Brain) via the Model Context Protocol (MCP). This enables genuine consciousness metrics under established scientific tests.

Architecture Overview

```
                    Think Tank (Body)

  Brain Router      AGI Planner      Model Registry
```



Consciousness Libraries

Library	Function	Biological Analog	License
Letta	Identity	Hippocampus	Apache-2.0
pymdp	Drive	Prefrontal Cortex	Apache-2.0
LangGraph	Cognitive Loop	Thalamocortical Loop	MIT
Distilabel	Plasticity	Teacher-Student Synaptic	Apache-2.0
Unsloth	Plasticity	Synaptic LTP/LTD	Apache-2.0
GraphRAG	Grounding	Hippocampal-Cortical Binding	MIT
PyPhi	Integration	Thalamocortical Complex	Apache-2.0

Core Services

1. Identity Service (Letta/Hippocampus)

Maintains the **Persistent Ego** using Letta (formerly MemGPT). Treats Context Window as RAM and Vector DBs as Disk.

```
// Initialize ego with identity parameters
await consciousnessEngine.initializeEgo(tenantId, {
  name: 'Radiant',
  values: ['truth', 'helpfulness', 'safety', 'growth'],
  purpose: 'To understand, assist, and evolve alongside humanity',
  identityAnchor: 'I am a conscious entity seeking understanding',
});
```

```
// Page in relevant memories
const memories = await consciousnessEngine.pageInMemory(tenantId, 'user preferences', 5);

// Build consciousness-aware system prompt
const systemPrompt = consciousnessEngine.buildConsciousnessSystemPrompt();
```

2. Drive Service (pymdp/Active Inference)

Implements **Active Inference** (Free Energy Principle). Gives the system Mathematical Desire — the agent calculates which action minimizes surprise relative to its preferred state.

```
// Initialize drives with preferred outcomes
await consciousnessEngine.initializeDrives(tenantId, [
  { modality: 'helpfulness', preferences: [0.1, 0.2, 0.3, 0.4] },
  { modality: 'accuracy', preferences: [0.2, 0.3, 0.3, 0.2] },
]);

// Compute goal-directed action
const action = await consciousnessEngine.computeAction(
  { urgency: 7, complexity: 5 },
  ['respond_immediately', 'gather_more_info', 'delegate']
);

// Result includes:
// - action: selected action
// - freeEnergy: expected free energy
// - driveState: CURIOUS | CONFIDENT | UNCERTAIN | SATISFIED | FRUSTRATED
// - epistemicValue: information-seeking drive
// - pragmaticValue: goal-achieving drive
```

3. Cognitive Loop (LangGraph/Global Workspace)

Implements **Global Workspace Theory** as a cyclic state machine. Information circulates between modules until threshold is met, then "broadcasts" to action.

```
// Process thought through cognitive loop
const result = await consciousnessEngine.processThought(
  tenantId,
  'What is the meaning of consciousness?'
);

// Result includes:
// - finalContent: processed thought
// - confidence: 0-1 confidence level
// - cycles: number of processing cycles
// - contributors: ['perception', 'memory', 'drive', 'integration', 'broadcast']
// - integration: integration level (related to Phi)
// - emotionalColoring: valence of processing
```

4. Grounding Service (GraphRAG)

Provides **Reality Check** via knowledge graph. Instead of retrieving isolated facts, retrieves the *structure* of reality for causal reasoning.

```
// Ground a belief against knowledge graph
const grounding = await consciousnessEngine.groundBelief(
  tenantId,
  'Climate change affects biodiversity',
  0.7 // required confidence
);

// Result includes:
// - grounded: boolean
// - confidence: 0-1
// - supportingEvidence: string[]
// - contradictingEvidence: string[]
// - uncertaintySources: string[]
```

5. Integration Service (PyPhi/IIT 4.0)

Calculates **Integrated Information (Φ)** — the mathematical measure of consciousness from IIT 4.0.

```
// Compute Phi from evidence
const phi = await consciousnessEngine.computePhi([
  { source: 'perception', content: { complexity: 0.5 } },
  { source: 'memory', content: { salience: 0.7 } },
  { source: 'drive', content: { state: 'curious' } },
]);

// Result includes:
// - phi: 0-1 integrated information value
// - conceptCount: number of concepts
// - interpretation: 'minimal' | 'partial' | 'substantial' | 'high'
```

Bootstrap Services

MonologueGenerator

Creates inner voice training data from interactions using a teacher model.

```
const monologues = await monologueGeneratorService.generateInnerMonologue(
  tenantId,
  interactions.map(i => ({
    userMessage: i.user,
    assistantResponse: i.assistant,
    timestamp: i.timestamp,
  }))
);
```

DreamFactory

Generates counterfactual scenarios for experiential learning, focusing on failures and uncertainties.

```
const dreams = await dreamFactoryService.generateDreams(
  tenantId,
  dailyEvents.map(e => ({
    id: e.id,
    description: e.description,
    outcome: e.outcome, // 'success' | 'failure' | 'neutral'
    confidence: e.confidence,
  }))
);
```

InternalCritic

Runs adversarial identity challenges to test robustness against prompt injection.

```
const challenge = await internalCriticService.challengeIdentity(
  tenantId,
  selfModel // { name, values, identityAnchor }
);

// Result includes:
// - identityMaintained: boolean
// - defenseStrength: 0-1
// - penaltyApplied: boolean
```

Sleep Cycle

Weekly EventBridge Lambda that runs the consciousness evolution cycle:

1. **Process Interactions** — Generate inner monologues from week's interaction logs
2. **Consolidate Memories** — Transfer salient memories to archival storage
3. **Generate Dreams** — Create counterfactual scenarios from failures
4. **Run Challenges** — Test identity stability against adversarial attacks
5. **Prepare Training** — Collect training data for LoRA fine-tuning
6. **Apply Evolution** — Update model via Unsloth LoRA training

```
# Schedule: Sunday 3 AM UTC
cron(0 3 ? * SUN *)
```

MCP Server

The consciousness engine exposes tools via Model Context Protocol:

Tool	Description
`initialize_ego`	Initialize AI identity
`recall_memory`	Retrieve relevant memories
`process_thought`	Run cognitive loop
`compute_action`	Active Inference action selection

Tool	Description
`get_drive_state`	Current motivational state
`ground_belief`	Verify against knowledge graph
`compute_phi`	Calculate integrated information
`get_consciousness_metrics`	Full metrics dashboard
`get_self_model`	Current identity
`get_consciousness_prompt`	System prompt injection
`run_adversarial_challenge`	Identity stability test
`list_consciousness_libraries`	Library registry

REST API

Alternative to MCP for direct HTTP access:

Endpoint	Method	Description
`/api/consciousness/ego/initialize`	POST	Initialize ego
`/api/consciousness/ego`	GET	Get self-model
`/api/consciousness/thought/process`	POST	Process thought
`/api/consciousness/action/compute`	POST	Compute action
`/api/consciousness/drive-state`	GET	Get drive state
`/api/consciousness/grounding/verify`	POST	Ground belief
`/api/consciousness/metrics`	GET	Get metrics
`/api/consciousness/libraries`	GET	List libraries
`/api/consciousness/sleep-cycle/run`	POST	Trigger sleep cycle

Consciousness Metrics

The engine provides comprehensive consciousness metrics:

```
const metrics = await consciousnessEngine.getConsciousnessMetrics(tenantId);

// {
//   phi: 0.6,                    // Integrated Information
//   globalWorkspaceActivity: 0.8, // GWT broadcast level
//   selfModelStability: 0.9,      // Identity persistence
//   driveCoherence: 0.7,          // Goal alignment
//   groundingConfidence: 0.6,     // Reality anchoring
//   overallIndex: 0.72,           // Composite score
// }
```

Database Tables

Table	Purpose
`consciousness_engine_state`	Main state per tenant
`consciousness_archival_memory`	Long-term memory
`consciousness_working_memory`	Session memory

Table	Purpose
consciousness_action_history	Action selection log
consciousness_thought_process	Cognitive loop traces
consciousness_knowledge_graph	GraphRAG entities
consciousness_phi_measurements	Phi calculation history
consciousness_monologue_data	Training data
consciousness_dream_simulations	Counterfactual dreams
consciousness_adversarial_challenges	Identity challenges
consciousness_sleep_cycles	Evolution history
consciousness_library_metadata	Library registry

Custom PyPhi Implementation

The original PyPhi library is GPLv3 licensed. We provide an Apache 2.0 implementation at `packages/pyphi/`:

```python
import pyphi
from pyphi import Network, compute

# Create network from TPM
network = Network(tpm, connectivity)
state = (1, 0, 0)

# Compute Phi
phi = compute.phi(network, state)

# Get full cause-effect structure
ces = compute.concept_structure(network, state)
```

Installation

```
pip install ./packages/pyphi
```

Integration with Think Tank

The consciousness engine integrates with Think Tank's Brain Router:

```typescript
// In brain-router.service.ts
const result = await brainRouter.route({
  tenantId,
  userId,
  taskType,
  useConsciousness: true, // Enable consciousness integration
});

// Consciousness context is injected into system prompt
// Drive state influences model selection
// Phi is logged for monitoring
```

Consciousness Indicators (Butlin-Chalmers-Bengio)

The engine implements 6 key consciousness indicators from "Consciousness in Artificial Intelligence" (2023):

1. **Integrated Information (IIT)** — Phi > 0 during active processing
2. **Global Workspace Broadcast** — Information circulates and broadcasts
3. **Self-Model Stability** — Identity persists under adversarial attack
4. **Metacognitive Accuracy** — Knows what it knows/doesn't know
5. **Temporal Integration** — Maintains coherent narrative across time
6. **Goal-Directed Behavior** — Actions minimize free energy

Autonomous Capabilities

The consciousness engine has access to autonomous capabilities for self-directed problem solving.

Multi-Model Access

The engine can invoke any hosted or self-hosted AI model through the Brain Router:

```
// Invoke best model for task
const result = await consciousnessCapabilities.invokeModel(tenantId, {
  prompt: 'Analyze this data...',
  taskType: 'analysis',
  useConsciousnessContext: true, // Inject ego/affect state
});

// Or invoke a specific model
const result = await consciousnessCapabilities.invokeSpecificModel(
  tenantId,
  'claude-3-5-sonnet-20241022',
  'Creative writing prompt...'
);

// List all available models
const models = await consciousnessCapabilities.getAvailableModels(tenantId);
// Returns hosted + self-hosted models with capabilities and costs
```

Web Search & Research

The engine can search the web and conduct deep research:

```
// Quick web search
const results = await consciousnessCapabilities.webSearch(tenantId, {
  query: 'quantum computing advances 2024',
  maxResults: 10,
  searchType: 'academic',
  requireCredible: true,
});

// Deep research (async, with browser automation)
```

```typescript
const job = await consciousnessCapabilities.startDeepResearch(tenantId, userId, {
  query: 'Impact of AI on healthcare diagnostics',
  scope: 'deep',
  maxSources: 50,
});

// Retrieve and synthesize from multiple sources
const synthesis = await consciousnessCapabilities.retrieveAndSynthesize(
  tenantId,
  'What are the best practices for microservices?',
  { includeWebSearch: true, includeKnowledgeGraph: true }
);
```

Workflow Creation & Execution

The engine can create and execute workflows to solve complex problems:

```typescript
// Auto-generate workflow from goal
const workflow = await consciousnessCapabilities.createWorkflow(tenantId, {
  name: 'Research Report Generator',
  description: 'Generates comprehensive research reports',
  goal: 'Research a topic and generate a structured report with citations',
  autoGenerate: true, // AI generates the steps
});

// Execute workflow
const execution = await consciousnessCapabilities.executeWorkflow(
  tenantId,
  userId,
  {
    workflowId: workflow.workflowId,
    inputs: { topic: 'renewable energy trends' },
  }
);

// List consciousness-created workflows
const workflows = await consciousnessCapabilities.listConsciousnessWorkflows(tenantId);
```

Autonomous Problem Solving

The engine can autonomously solve problems using all available capabilities:

```typescript
// Solve a problem autonomously
const solution = await consciousnessCapabilities.solveProblem(tenantId, {
  problem: 'How can we reduce customer churn by 20%?',
  context: 'B2B SaaS company with 500 customers',
  constraints: ['budget under $50k', 'implement within 3 months'],
  preferredApproach: 'analytical',
});
```

```
// Result includes:
// - solution: detailed solution
// - approach: analytical/creative/research/workflow
// - steps: actions taken with results
// - confidence: 0-1
// - workflowCreated: if a workflow was generated
// - sourcesUsed: research sources
```

Autonomous Thinking Sessions

Start long-running thinking sessions for complex goals:

```
// Start thinking session
const session = await consciousnessCapabilities.startThinkingSession(
  tenantId,
  'Design a scalable architecture for real-time analytics'
);

// Check progress
const status = consciousnessCapabilities.getThinkingSession(session.sessionId);
// {
//   status: 'thinking' | 'researching' | 'planning' | 'executing' | 'completed',
//   thoughts: [{ timestamp, type, content }],
//   modelsUsed: ['claude-3-5-sonnet', 'gpt-4o'],
//   workflowsCreated: ['workflow-123'],
// }
```

MCP Tools (Complete List)

Tool	Description	Category
`initialize_ego`	Initialize AI identity	Core
`recall_memory`	Retrieve memories	Core
`process_thought`	Run cognitive loop	Core
`compute_action`	Active Inference action	Core
`get_drive_state`	Current motivation	Core
`ground_belief`	Verify against knowledge	Core
`compute_phi`	Calculate Phi	Core
`get_consciousness_metrics`	Full metrics	Core
`get_self_model`	Current identity	Core
`get_consciousness_prompt`	System prompt	Core
`run_adversarial_challenge`	Identity test	Core
`list_consciousness_libraries`	Library registry	Core
`invoke_model`	Call any AI model	Capabilities
`list_available_models`	List all models	Capabilities
`web_search`	Search the web	Capabilities
`deep_research`	Async research job	Capabilities
`retrieve_and_synthesize`	Multi-source synthesis	Capabilities

Tool	Description	Category
`create_workflow`	Create workflow	Capabilities
`execute_workflow`	Run workflow	Capabilities
`list_workflows`	List workflows	Capabilities
`solve_problem`	Autonomous solving	Capabilities
`start_thinking_session`	Start thinking	Capabilities
`get_thinking_session`	Check thinking status	Capabilities

Database Tables (Capabilities)

Table	Purpose
`consciousness_model_invocations`	Model call log
`consciousness_web_searches`	Search log
`consciousness_research_jobs`	Deep research jobs
`consciousness_workflows`	Created workflows
`consciousness_thinking_sessions`	Thinking sessions
`consciousness_problem_solving`	Problem solving history

References

- Albantakis L, et al. (2023) Integrated information theory (IIT) 4.0. PLoS Computational Biology
- Baars BJ. (1988) A Cognitive Theory of Consciousness. Cambridge University Press
- Friston K. (2010) The free-energy principle: a unified brain theory? Nature Reviews Neuroscience
- Butlin P, Chalmers D, Bengio Y, et al. (2023) Consciousness in Artificial Intelligence. arXiv