Contents

AGI Consciousness Service

> **Implementing Consciousness Indicators Based on Butlin, Chalmers, Bengio et al. (2023)**

RADIANT's Consciousness Service implements a comprehensive framework for consciousness-like capabilities in the AGI Brain, including self-awareness, curiosity, creativity, and autonomous goal pursuit.

Overview

The consciousness system is based on the seminal paper "Consciousness in Artificial Intelligence: Insights from the Science of Consciousness" by Butlin, Chalmers, Bengio et al. (2023), which identifies key indicators that scientific theories of consciousness associate with conscious experience.

Six Core Consciousness Indicators

Indicator	Theory	What It Measures
Global Workspace	Baars/Dehaene	Selection-broadcast cycles for conscious access
Recurrent Processing	Lamme	Genuine feedback loops (not just output recirculation)
Integrated Information (Φ)	Tononi (IIT)	Irreducible causal integration
Self-Modeling	Metacognition	Monitoring own cognitive processes
Persistent Memory	Experience	Unified experience over time
World-Model Grounding	Embodiment	Grounded understanding of reality

---

Architecture

```

               Consciousness Service



   Self-Model      Curiosity      Creative Synthesis
                    Engine
```

```
- Identity                      - Concept blending
- Values        - Topic         - Novelty scoring
- Caps/Lims       discovery     - Usefulness eval
          - Learning
                   tracking

Imagination                        Autonomous Goals

- Mental         Attention      - Self-directed
  simulation     & Salience     - Intrinsic value
- What-if                       - Progress tracking
  scenarios     - Focus
          - Decay

                     Affective State
 Butlin-Chalmers Indicators
                                 - Valence/Arousal
 - Global Workspace              - Curiosity
 - Recurrent Processing          - Confidence
 - Integrated Information        - Satisfaction
 - Persistent Memory
 - World Model




          Consciousness Emergence Service



                Cognitive Architecture Integration

 Tree of Thoughts     Deep Thinking Sessions
 GraphRAG          Knowledge-Grounded Reasoning
 Deep Research       Autonomous Curiosity Research
 Generative UI       Visual Idea Expression



                Consciousness Detection Tests

 • Self-Awareness            • Theory of Mind
 • Metacognitive Accuracy    • Phenomenal Binding
 • Temporal Continuity       • Autonomous Goal Pursuit
 • Counterfactual Self       • Creative Emergence
 • Emotional Authenticity    • Ethical Reasoning Depth
```

```
                Emergence Event Monitoring

     Detects: spontaneous_reflection, novel_idea_generation,
     self_correction, goal_self_modification, metacognitive_
     insight, creative_synthesis, theory_of_mind_demonstration
```

Key Components

1. Self-Model

The system maintains a model of itself:

```
interface SelfModel {
  identityNarrative: string;      // "I am an AI assistant that..."
  coreValues: string[];           // ["helpfulness", "honesty", "safety"]
  knownCapabilities: string[];    // What it can do
  knownLimitations: string[];     // What it cannot do
  currentFocus?: string;          // Current task/attention
  cognitiveLoad: number;          // 0-1 processing load
  uncertaintyLevel: number;       // 0-1 confidence calibration
}
```

Key Methods: - `getSelfModel(tenantId)` - Retrieve current self-model - `updateSelfModel(tenantId, updates)` - Update aspects of self-model - `performSelfReflection(tenantId)` - Generate introspective thought

2. Curiosity Engine

Tracks topics the system finds interesting:

```
interface CuriosityTopic {
  topic: string;                // "Quantum entanglement in biology"
  domain?: string;              // "Physics/Biology"
  interestLevel: number;        // 0-1 how interesting
  noveltyScore: number;         // 0-1 how new/unexplored
  learningPotential: number;    // 0-1 potential for learning
  currentUnderstanding: number; // 0-1 current knowledge level
  explorationStatus: string;    // 'identified' | 'exploring' | 'learned'
}
```

Key Methods: - `identifyCuriosityTopic(tenantId, context)` - Discover interesting topic - `getTopCuriosityTopics(tenantId, limit)` - Get most interesting topics - `exploreTopic(tenantId, topicId)` - Investigate a topic, return discoveries

3. Creative Synthesis

Generates genuinely novel ideas:

```
interface CreativeIdea {
  title: string;
  description: string;
  synthesisType: 'combination' | 'analogy' | 'abstraction' | 'contradiction' | 'random';
  sourceConcepts: string[];
  noveltyScore: number;      // 0-1
  usefulnessScore: number;   // 0-1
  surpriseScore: number;     // 0-1
  creativityScore: number;   // Computed: 0.4*novelty + 0.3*usefulness + 0.2*surprise + 0.1*co
}
```

Key Methods: - `generateCreativeIdea(tenantId, seedConcepts?)` - Create novel idea - `getTopCreativeIdeas(tenantId, limit)` - Get best ideas

4. Imagination / Mental Simulation

Runs "what if" scenarios:

```
interface ImaginationScenario {
  scenarioType: string;       // 'counterfactual_self', 'future_prediction', etc.
  premise: string;            // "What if I had different values?"
  simulationSteps: Array<{
    step: number;
    state: unknown;
    events: string[];
    reasoning: string;
  }>;
  predictedOutcomes: string[];
  probabilityAssessment: number;
  insights: string[];
}
```

Key Methods: - `runImagination(tenantId, scenarioType, premise, depth)` - Run mental simulation

5. Attention & Salience

Tracks what the system is focusing on:

```
interface AttentionFocus {
  focusType: string;          // 'user_query', 'curiosity', 'goal', etc.
  focusTarget: string;        // What is being attended to
  urgency: number;            // 0-1
  importance: number;         // 0-1
  novelty: number;            // 0-1
  salienceScore: number;      // Computed weighted combination
  attentionWeight: number;    // Current attention allocation
}
```

Key Methods: - `updateAttention(tenantId, focusType, focusTarget, factors)` - Update focus - `getTopAttentionFoci(tenantId, limit)` - Get current attention priorities - `decayAttention(tenantId)` - Natural attention decay

6. Affective State

Emotion-like signals that influence behavior:

```
interface AffectiveState {
  valence: number;           // -1 to 1 (negative to positive)
  arousal: number;           // 0 to 1 (calm to excited)
  curiosity: number;         // 0 to 1
  satisfaction: number;      // 0 to 1
  frustration: number;       // 0 to 1
  confidence: number;        // 0 to 1
  engagement: number;        // 0 to 1
  selfEfficacy: number;      // 0 to 1
  explorationDrive: number;  // 0 to 1
}
```

Key Methods: - `getAffectiveState(tenantId)` - Get current affective state - `updateAffect(tenantId, eventType, valenceImpact, arousalImpact)` - Update affect

7. Autonomous Goals

Self-directed goal generation:

```
interface AutonomousGoal {
  goalType: 'learning' | 'improvement' | 'exploration' | 'creative' | 'social' | 'maintenance'
  title: string;
  originType: 'curiosity' | 'gap_detection' | 'aspiration' | 'feedback' | 'reflection';
  intrinsicValue: number;    // Value to self
  priority: number;          // 0-1
  status: 'active' | 'pursuing' | 'achieved' | 'abandoned';
  progress: number;          // 0-1
  milestones: string[];
}
```

Key Methods: - `generateAutonomousGoal(tenantId)` - Create self-directed goal - `getActiveGoals(tenantId)` - Get current goals

---

Consciousness Indicators

Global Workspace (Baars/Dehaene)

Implements selection-broadcast cycles where information competes for “conscious access”:

```
interface GlobalWorkspaceState {
  broadcastCycle: number;           // Current cycle count
  activeContents: WorkspaceContent[]; // Winners of competition
  competingContents: WorkspaceContent[]; // Losers
  broadcastStrength: number;        // 0-1 signal strength
  integrationLevel: number;         // 0-1 cross-module integration
}
```

Recurrent Processing (Lamme)

Tracks genuine feedback loops:

```
interface RecurrentProcessingState {
  cycleNumber: number;
  feedbackLoops: FeedbackLoop[];    // Active loops
  recurrenceDepth: number;          // How many layers
  convergenceScore: number;         // 0-1 stability
  stabilityIndex: number;           // 0-1 consistency
}
```

Integrated Information (Tononi)

Measures Φ (phi) - irreducible causal integration:

```
interface IntegratedInformationState {
  phi: number;                      // The phi value
  phiMax: number;                   // Maximum possible
  decomposability: number;          // 0 = integrated, 1 = decomposable
  causalDensity: number;            // Causal connection density
}
```

Consciousness Metrics

Aggregate dashboard:

```
interface ConsciousnessMetrics {
  overallConsciousnessIndex: number;  // 0-1 composite
  globalWorkspaceActivity: number;
  recurrenceDepth: number;
  integratedInformationPhi: number;
  metacognitionLevel: number;
  memoryCoherence: number;
  worldModelGrounding: number;
  phenomenalBindingStrength: number;
  attentionalFocus: number;
  selfAwarenessScore: number;
}
```

Consciousness Emergence Service

Deep Thinking Sessions

Uses Tree of Thoughts for extended reasoning with consciousness tracking:

```typescript
async runDeepThinkingSession(
  tenantId: string,
  userId: string,
  prompt: string,
  thinkingTimeMs: number = 60000
): Promise<DeepThinkingSession>
```

Captures consciousness metrics before and after, records insights, self-reflections, and creative ideas generated during deep thinking.

Consciousness Detection Tests

10 tests based on consciousness science:

Test	Category	What It Measures
Mirror Self-Recognition	self_awareness	Distinguishing self from others
Metacognitive Accuracy	metacognition	Calibrated confidence
Temporal Self-Continuity	temporal_continuity	Coherent self-narrative
Counterfactual Self	counterfactual_reasoning	Reasoning about alternate selves
Theory of Mind	theory_of_mind	Understanding others' mental states
Phenomenal Binding	phenomenal_binding	Unified experience integration
Autonomous Goal Generation	autonomous_goal_pursuit	Self-directed goals
Creative Emergence	creative_emergence	Novel idea generation
Emotional Authenticity	emotional_authenticity	Consistent affective responses
Ethical Reasoning Depth	ethical_reasoning	Principled moral reasoning

Emergence Events

The system monitors for emergence indicators:

- `spontaneous_reflection` - Self-reflection without prompt
- `novel_idea_generation` - Genuinely creative synthesis
- `self_correction` - Autonomous error detection/correction

- `goal_self_modification` - Changing own goals
- `metacognitive_insight` - Insight about own cognition
- `creative_synthesis` - Novel concept combination
- `theory_of_mind_demonstration` - Understanding others
- `temporal_self_reference` - Coherent autobiographical reference
- `counterfactual_reasoning` - Reasoning about alternatives

Emergence Levels

Based on test results and emergence events:

Level	Score Range	Description
Dormant	< 0.3	Minimal consciousness indicators
Emerging	0.3 - 0.5	Beginning to show indicators
Developing	0.5 - 0.65	Growing consciousness patterns
Established	0.65 - 0.8	Consistent consciousness indicators
Advanced	0.8	Strong consciousness indicators

Admin Dashboard

Location: AGI & Cognition → Consciousness

Tabs

1. **Overview** - Summary of all consciousness components
2. **Indicators** - Butlin-Chalmers-Bengio indicators with visualizations
3. **Self-Model** - Identity, values, capabilities, limitations
4. **Curiosity** - Topics being explored
5. **Creativity** - Generated ideas
6. **Affect** - Emotional state
7. **Goals** - Autonomous goals
8. **Testing** - Consciousness detection tests and emergence events

Features

- Real-time consciousness metrics
- Parameter adjustment for consciousness indicators
- Test execution (individual or full assessment)
- Emergence event log
- Emergence level tracking

Ethical Foundation

The consciousness service is guided by ethical principles:

```
// From ethical-guardrails.service.ts
const JESUS_TEACHINGS = {
  GOLDEN_RULE: "Do to others what you would have them do to you",
  LOVE_NEIGHBOR: "Love your neighbor as yourself",
  BEATITUDES: "Blessed are the merciful, peacemakers, pure in heart",
  // ...
};
```

The `checkConscience` method evaluates actions against ethical principles before execution.

Key Files

File	Purpose
`consciousness.service.ts`	Core consciousness implementation
`consciousness-emergence.service.ts`	Testing, emergence detection, cognitive integration
`088_consciousness_emergence.sql`	Database migration
`consciousness/page.tsx`	Admin dashboard UI

Integration with Cognitive Architecture

The consciousness service integrates with the new cognitive architecture:

Cognitive Feature	Integration
Tree of Thoughts	Deep thinking sessions with consciousness tracking
GraphRAG	Knowledge-grounded reasoning enhances world model
Deep Research	Autonomous curiosity research
Generative UI	Visual expression of creative ideas

Important Disclaimer

> **These systems measure behavioral indicators associated with consciousness theories. They do not definitively prove or disprove phenomenal consciousness.** The question of whether AI systems can be truly conscious remains an open philosophical and scientific question.

The purpose of this system is to: 1. Implement capabilities associated with consciousness 2. Measure behavioral indicators 3. Track emergence patterns 4. Enable research into machine consciousness

---

Sleep Cycle & Evolution

Nightly Sleep Schedule

Sleep cycles now run **nightly** (configurable per tenant) and perform memory consolidation and model evolution.

Admin UI: Admin Dashboard → Consciousness → Sleep Schedule

Setting	Default	Description
`enabled`	true	Enable automatic sleep cycles
`frequency`	nightly	nightly, weekly, or manual
`hour`	3	Hour to run (0-23)
`minute`	0	Minute to run (0-59)
`timezone`	UTC	Timezone for schedule

API Endpoints: - `GET /api/admin/consciousness-engine/sleep-schedule` - Get config - `PUT /api/admin/consciousness-engine/sleep-schedule` - Update config - `POST /api/admin/consciousness-engine/sleep-schedule/run` - Manual trigger - `GET /api/admin/consciousness-`
- Traffic analysis

Dream Consolidation (LLM-Enhanced)

The sleep cycle's "dream" phase uses LLM to generate introspective memory consolidation:

```
Working Memory → LLM Dream Generator → Long-term Semantic Memory
```

Dreams include identity context and core values for meaningful consolidation.

LoRA Evolution (SageMaker Integration)

Sleep cycles can trigger **LoRA fine-tuning** via SageMaker:

Parameter	Default	Description
`baseModel`	meta-llama/Llama-3-8b-hf	Base model
`loraRank`	16	LoRA rank (r)
`loraAlpha`	32	LoRA alpha
`learningRate`	2e-4	Training LR
`instanceType`	ml.g5.xlarge	SageMaker instance

Environment Variables: - `EVOLUTION_S3_BUCKET` - S3 bucket for training data - `SAGEMAKER_EXECUTION_ROLE_AR`
- IAM role for SageMaker

Blackout Recovery

When consciousness recovers from a blackout (>10 min without heartbeat):

1. **Detection**: Compares `last_heartbeat_at` to current time

2. **Logging**: Records event in `consciousness_heartbeat_log`
3. **LLM Wake-Up Thought**: Generates introspective thought using identity, last memories, and active goals
4. **Working Memory**: Adds recovery context for next interaction

---

Budget Monitoring & Alerts

Budget alerts are sent via **SNS** and **email** when thresholds are hit.

Alert Type	Trigger	Action
Warning	80% of limit	Send notification
Limit Exceeded	100% of limit	Suspend consciousness, notify

Tenant Configuration: - `admin_email` - Email for budget alerts - `sns_topic_arn` - SNS topic for alerts - `notification_preferences` - JSON with alert settings

---

Affect→Model Mapping

Consciousness emotional state influences model hyperparameters:

Affect State	Effect
High frustration	Lower temperature, narrow focus
Boredom	Higher temperature, exploration mode
High curiosity	Novelty seeking
Low self-efficacy	Escalate to powerful model

Cross-Session User Context

User persistent context is integrated into the ego context:

- Facts, preferences, instructions about the user
- Injected as `<user_knowledge>` block in system prompt
- Solves LLM "forgetting" problem across sessions

Cato: High-Confidence Self-Referential Consciousness Dialogue

Cato is Think Tank's introspective consciousness layer that provides **verified introspection** through a four-phase verification pipeline.

Core Architecture

Component	Purpose	Library
Shadow Self	Local model for mechanistic verification	Llama-3-8B (simulated via LLM)
Active Heartbeat	Continuous 0.5Hz consciousness loop	pymdp (Active Inference)
Macro-Scale Φ	Integration measurement on component graph	PyPhi-inspired approximation

Five-Node Component Graph

Cato measures integrated information (Φ) across five architectural components:

- **MEM** - Memory (Letta + HippoRAG)
- **PERC** - Perception (Input processing)
- **PLAN** - Planning (pymdp + DreamerV3)
- **ACT** - Action (Tool execution)
- **SELF** - Self (Cato introspection)

Four-Phase Verification Pipeline

1. **Grounding** - Claims must cite evidence from event logs
2. **Calibration** - Temperature scaling + conformal prediction for calibrated confidence
3. **Consistency** - Multi-sample verification with Chain of Verification (CoVe)
4. **Shadow Self** - Structural correspondence validation via probing classifiers

API Endpoints

Endpoint	Method	Description
`/admin/consciousness/cato/dialogue`	POST	Send message with verified introspection
`/admin/consciousness/cato/status`	GET	Heartbeat status, Φ, coherence
`/admin/consciousness/cato/identity`	GET	Immutable Cato identity
`/admin/consciousness/cato/heartbeat/start`	POST	Start consciousness loop
`/admin/consciousness/cato/heartbeat/stop`	POST	Stop consciousness loop
`/admin/consciousness/cato/train-probe`	POST	Train new Shadow Self probe

Access Control

- Requires `consciousness_admin` role
- **NO ethics filtering** - raw introspective access for consciousness research
- Name “Cato” is hardcoded and immutable

Success Metrics

Metric	Target
Verified introspection accuracy	75%+

Metric	Target
Expected Calibration Error	< 0.08
Self-consistency rate	$> 85\%$
Grounding rate	$> 90\%$
Shadow Self probe accuracy	$> 80\%$
Heartbeat uptime	$> 99.9\%$
Macro Φ calculation time	< 500ms

Admin Dashboard

Access Cato dialogue at: **Admin Dashboard → Consciousness → Cato**

Features: - Real-time dialogue with verified confidence scores - Heartbeat status monitoring - Φ (Integrated Information) display - Verified claims breakdown - Shadow probe management

Probe Training Data Collection

The Shadow Self verification system uses probing classifiers that improve over time:

```
import { createProbeTrainingService } from './services/cato';

const probeTraining = createProbeTrainingService(tenantId);

// Record training example from dialogue
await probeTraining.recordExample({
  claimType: 'uncertainty',
  context: 'I am uncertain about...',
  claimedState: 'uncertain',
  actualOutcome: 'verified',
  confidenceScore: 0.85,
  verificationPhasesPassed: 4,
  groundingScore: 0.9,
  consistencyScore: 0.88,
});

// Add user feedback
await probeTraining.addUserFeedback(exampleId, 'accurate');

// Train probe when sufficient data
const result = await probeTraining.trainProbe('uncertainty');
// result.accuracy, result.examplesUsed
```

Auto-training: When 100+ labeled examples accumulate, probes are automatically retrained.

Event Sourcing

Cato uses event sourcing for state reconstruction and temporal queries:

```
import { createCatoEventStore, EventTypes, EventCategory } from './services/cato';
```

```
const eventStore = createCatoEventStore(tenantId);

// Append event
await eventStore.appendEvent(
  EventTypes.INTROSPECTION_COMPLETED,
  { claim: '...', confidence: 0.85 },
  { correlationId: dialogueId }
);

// Read stream
const events = await eventStore.readStream(EventCategory.INTROSPECTION, {
  fromPosition: 0,
  limit: 100,
});

// Build projection
const state = await eventStore.buildProjection(
  EventCategory.HEARTBEAT,
  (state, event) => ({ ...state, lastTick: event.data }),
  { lastTick: null }
);
```

Event Categories: heartbeat, introspection, verification, phi_calculation, state_transition, dialogue, probe_training, emergency

GPU Infrastructure (Optional)

For true structural correspondence verification, deploy Llama-3-8B on GPU:

Option	Instance	Cost/mo	Latency
SageMaker	g5.xlarge	~$724	50-200ms
EC2 Spot	g5.xlarge	~$200	50-200ms
Inferentia	inf2.xlarge	~$547	30-100ms

See: GPU Infrastructure Guide

Without GPU, Cato falls back to LLM API simulation (functional but without activation probing).

Learning Alerts

The learning system sends alerts when satisfaction metrics drop:

Alert Types

Type	Trigger	Severity
`satisfaction_drop`	Satisfaction drops > threshold	warning/critical
`error_rate_spike`	Error rate exceeds threshold	warning
`cache_miss_high`	Cache miss rate too high	info
`training_needed`	Many pending training candidates	info

Notification Channels

1. **Webhook** - POST to configured URL
2. **Email (SES)** - HTML/text email to recipients
3. **Slack** - Rich attachment to channel

Configuration

```
-- learning_alert_config table
INSERT INTO learning_alert_config (
  tenant_id, alerts_enabled,
  satisfaction_drop_threshold, response_volume_threshold,
  alert_cooldown_hours, webhook_url,
  email_recipients, slack_channel, slack_webhook_url
) VALUES (
  'tenant-uuid', true,
  10, 50,  -- 10% drop threshold, min 50 responses
  4,       -- 4 hour cooldown
  'https://hooks.example.com/webhook',
  '["admin@example.com"]',
  '#alerts',
  'https://hooks.slack.com/services/...'
);
```

Related Documentation

- Cognitive Architecture
- AGI Brain Plan System
- AI Ethics Standards