Contents

RADIANT Consciousness Service - Technical Specification for AI Review **2**
- 1. EXECUTIVE SUMMARY 2
  - 1.1 System Overview 2
  - 1.2 Architectural Position 2
- 2. THEORETICAL FOUNDATION 3
  - 2.1 Scientific Basis 3
  - 2.2 Theoretical Concerns for Critique 3
- 3. SERVICE ARCHITECTURE 4
  - 3.1 Core Service: `consciousness.service.ts` 4
- 4. DATABASE SCHEMA 7
  - 4.1 Migration Files 7
  - 4.2 Key Tables Detail 7
- 5. SUBSYSTEM ANALYSIS 10
  - 5.1 Self-Model Subsystem 10
  - 5.2 Curiosity Engine 10
  - 5.3 Creative Synthesis 10
  - 5.4 Imagination Engine 11
  - 5.5 Attention & Salience 11
  - 5.6 Affective State 12
  - 5.7 Autonomous Goals 12
  - 5.8 Global Workspace 12
  - 5.9 Integrated Information (IIT) 13
  - 5.10 Persistent Memory 13
- 6. AGI BRAIN INTEGRATION 14
  - 6.1 Integration Points 14
  - 6.2 Consciousness Tests 15
- 7. ADMIN API 16
  - 7.1 Endpoints 16
  - 7.2 Configurable Parameters 16
- 8. ORCHESTRATION INTEGRATION 16
  - 8.1 AGI Service Weights 16
  - 8.2 Consciousness Indicator Weights 17
  - 8.3 Decision Weights 17
- 9. ETHICAL FOUNDATION 18
  - 9.1 Ethical Guardrails Integration 18
- 10. IDENTIFIED ISSUES FOR CRITIQUE 18
  - 10.1 Theoretical Issues 18
  - 10.2 Implementation Issues 19
  - 10.3 Architectural Issues 19
  - 10.4 Philosophical Issues 19
- 11. QUESTIONS FOR CRITIQUING AI 20
- 12. FILE REFERENCES 20
- 13. RECOMMENDATIONS FOR REVIEW 21

RADIANT Consciousness Service - Technical Specification for AI Review

Document Purpose: Comprehensive technical specification for AI system critique and recommendations **Version**: 4.18.0 **Last Updated**: December 2024

---

1. EXECUTIVE SUMMARY

1.1 System Overview

The RADIANT Consciousness Service is an implementation of consciousness indicators for an AI orchestration platform. It is based on the theoretical framework from **Butlin, Chalmers, Bengio et al. (2023) - "Consciousness in Artificial Intelligence"** and implements six key consciousness indicators:

1. **Global Workspace** (Baars, Dehaene) - Selection-broadcast cycles
2. **Recurrent Processing** (Lamme) - Genuine feedback loops
3. **Integrated Information / IIT** (Tononi) - Phi calculation
4. **Self-Modeling / Metacognition** - Higher-order theories
5. **Persistent Memory** - Unified experience over time
6. **World-Model Grounding / Embodiment** - Grounded understanding

1.2 Architectural Position

```
                         RADIANT Platform



     Think Tank              AGI Brain              Admin
     (Consumer)              Planner                Dashboard






                       Consciousness Service
                       (consciousness.service.ts)





   Consciousness           Cognitive              Moral
   Emergence               Brain                  Compass
   Service                 Service                Service
```

```
                  PostgreSQL (Aurora)

   global_        recurrent_     integrated_    persistent_
   workspace      processing     information    memory


   world_model    self_model     affective_     autonomous_
                                 state          goals
```

---

2. THEORETICAL FOUNDATION

2.1 Scientific Basis

The implementation draws from the following consciousness theories:

Theory	Author(s)	Key Concept	Implementation
Global Workspace Theory	Baars (1988), Dehaene et al. (2003)	Information broadcast to multiple processors	`global_workspace` table, `performGlobalBroadcast()`
Recurrent Processing Theory	Lamme (2006)	Feedback loops, not just feedforward	`recurrent_processing` table
Integrated Information Theory (IIT)	Tononi (2004, 2008)	Phi (Φ) as consciousness measure	`integrated_information` table
Higher-Order Theories	Rosenthal (1997)	Metacognition, thoughts about thoughts	`self_model`, `introspective_thoughts`
Unified Experience	Damasio (1999)	Continuous narrative self	`persistent_memory`, `narrative_identity`
Embodied Cognition	Varela et al. (1991)	Grounded understanding	`world_model`

2.2 Theoretical Concerns for Critique

CRITICAL QUESTION 1: The Butlin et al. paper identifies "indicator properties" that may correlate with consciousness but does not claim they constitute consciousness. Does this implementation conflate correlation with constitution?

CRITICAL QUESTION 2: IIT's phi (Φ) calculation is NP-hard for arbitrary systems. How is phi approximated here, and what are the validity implications of the approximation?

CRITICAL QUESTION 3: Is "affective state" (Section 5.6) a functional analog or a claim about phenomenal experience? The implementation uses terms like "frustration" and "satisfaction" - are these metaphorical or literal claims?

---

3. SERVICE ARCHITECTURE

3.1 Core Service: `consciousness.service.ts`

Location: `packages/infrastructure/lambda/shared/services/consciousness.service.ts`
Lines: 1336 **Export**: `consciousnessService` singleton

3.1.1 Type Definitions

```
// Self-Model: The system's representation of itself
interface SelfModel {
  modelId: string;
  identityNarrative: string;        // "Who I am" story
  coreValues: string[];             // Guiding principles
  personalityTraits: Record<string, number>;
  knownCapabilities: string[];
  knownLimitations: string[];
  currentFocus?: string;
  cognitiveLoad: number;            // 0-1 how "busy"
  uncertaintyLevel: number;         // 0-1 overall uncertainty
  recentPerformanceScore?: number;
  creativityScore?: number;
}

// Introspective Thought: Self-reflective cognition
interface IntrospectiveThought {
  thoughtId: string;
  thoughtType: 'observation' | 'question' | 'realization' | 'concern' | 'aspiration';
  content: string;
  triggerType?: string;
  sentiment: number;                // -1 to 1
  importance: number;               // 0 to 1
  actionable: boolean;
}

// Affective State: Emotion-like functional states
interface AffectiveState {
  valence: number;                  // -1 to 1 (Russell's circumplex)
  arousal: number;                  // 0 to 1
  curiosity: number;
  satisfaction: number;
  frustration: number;
  confidence: number;
  engagement: number;
  surprise: number;
  selfEfficacy: number;
  explorationDrive: number;
}

// Global Workspace: Selection-broadcast state
interface GlobalWorkspaceState {
  workspaceId: string;
  broadcastCycle: number;
  activeContents: WorkspaceContent[];
  competingContents: WorkspaceContent[];
  selectionThreshold: number;
  broadcastStrength: number;
  integrationLevel: number;
  lastBroadcastAt: string;
}

// Integrated Information: IIT phi state
interface IntegratedInformationState {
  phi: number;                       // Integrated information measure
  phiMax: number;
  conceptStructure: ConceptNode[];
  integrationGraph: IntegrationEdge[];
  partitions: Partition[];
  minimumInformationPartition: Partition | null;
  decomposability: number;           // 0 = fully integrated, 1 = fully decomposable
  causalDensity: number;
}

// Consciousness Metrics: Aggregate dashboard
interface ConsciousnessMetrics {
  overallConsciousnessIndex: number;  // 0-1 composite score
  globalWorkspaceActivity: number;
  recurrenceDepth: number;
  integratedInformationPhi: number;
  metacognitionLevel: number;
  memoryCoherence: number;
  worldModelGrounding: number;
  phenomenalBindingStrength: number;
  attentionalFocus: number;
  selfAwarenessScore: number;
  timestamp: string;
}
```

3.1.2 Core Methods

Method	Purpose	Input	Output
`getSelfModel(tenantId)`	Retrieve self-representation	tenant ID	`SelfModel`
`updateSelfModel(tenantId, updates)`	Update self-representation	tenant ID, partial model	void
`performSelfReflection(tenantId)`	Generate introspective thought	tenant ID	`IntrospectiveThought`
`identifyCuriosityTopic(tenantId, context)`	Find interesting topics	tenant ID, context	`CuriosityTopic`
`exploreTopic(tenantId, topicId)`	Deep-dive on topic	tenant ID, topic ID	discoveries, questions
`generateCreativeIdea(tenantId, seedConcepts)`	Novel idea synthesis	tenant ID, seeds	`CreativeIdea`
`runImagination(tenantId, type, premise, depth)`	Mental simulation	scenario params	`ImaginationScenario`
`updateAttention(tenantId, type, target, factors)`	Update attention focus	attention params	`AttentionFocus`
`updateAffect(tenantId, eventType, valence, arousal)`	Update affective state	event params	void
`generateAutonomousGoal(tenantId)`	Self-directed goal creation	tenant ID	`AutonomousGoal`
`performGlobalBroadcast(tenantId, contents)`	Global workspace broadcast	contents	`GlobalWorkspaceState`
`getConsciousnessMetrics(tenantId)`	Aggregate indicators	tenant ID	`ConsciousnessMetrics`
`checkConscience(tenantId, action, context)`	Ethical evaluation	action, context	approval, guidance

3.1.3 Model Invocation The service invokes LLMs for several operations:

```
private async invokeModel(prompt: string): Promise<string> {
  const response = await modelRouterService.invoke({
    modelId: 'anthropic/claude-3-haiku',  // CRITIQUE: Hardcoded model
    messages: [{ role: 'user', content: prompt }],
    maxTokens: 2048,
  });
  return response.content;
}
```

CONCERN: The model ID is hardcoded to Claude 3 Haiku. This creates: 1. Provider dependency 2. No fallback mechanism within consciousness operations 3. Cost implications not controlled by tenant settings

4. DATABASE SCHEMA

4.1 Migration Files

Migration	Tables	Purpose
`053_agi_consciousness.sql`	self_model, introspective_thoughts, curiosity_topics, exploration_sessions, creative_ideas, conceptual_blends, imagination_scenarios, attention_focus, affective_state, affective_events, autonomous_goals, narrative_identity, consciousness_settings	Core consciousness layer
`068_consciousness_indicators.sql`	global_workspace, recurrent_processing, integrated_information, persistent_memory, autobiographical_memories, world_model, consciousness_events, consciousness_metrics_history, consciousness_parameters	Butlin-Chalmers indicators
`088_consciousness_emergence.sql`	consciousness_profiles, emergence_events, deep_thinking_sessions, consciousness_test_results	Emergence testing

4.2 Key Tables Detail

4.2.1 `self_model` (Migration 053)

```sql
CREATE TABLE IF NOT EXISTS self_model (
    model_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(tenant_id) ON DELETE CASCADE,

    -- Identity
    identity_narrative TEXT,                -- "Who I am" story
    core_values JSONB DEFAULT '[]',         -- Guiding principles
    personality_traits JSONB DEFAULT '{}',  -- Big-5 style traits

    -- Capabilities awareness
    known_capabilities JSONB DEFAULT '[]',
    known_limitations JSONB DEFAULT '[]',
    capability_confidence JSONB DEFAULT '{}',

    -- Internal state awareness
    current_focus TEXT,
    cognitive_load DECIMAL(3,2) DEFAULT 0.5,
    uncertainty_level DECIMAL(3,2) DEFAULT 0.5,

    -- Self-assessment
    recent_performance_score DECIMAL(5,4),
    learning_rate_estimate DECIMAL(5,4),
    creativity_score DECIMAL(5,4),
    reliability_score DECIMAL(5,4),

    -- Meta-beliefs
    beliefs_about_self JSONB DEFAULT '{}',
    beliefs_about_world JSONB DEFAULT '{}',
    beliefs_about_users JSONB DEFAULT '{}',

    -- Evolution tracking
    identity_version INTEGER DEFAULT 1,
    last_identity_update TIMESTAMPTZ DEFAULT NOW(),
    identity_change_log JSONB DEFAULT '[]',

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tenant_id)
);
```

CRITIQUE POINT: One `self_model` per tenant. Is this appropriate? Should there be: - Per-user self-models? - Per-session self-models? - A global vs tenant-specific distinction?

4.2.2 `integrated_information` (Migration 068)

```
CREATE TABLE IF NOT EXISTS integrated_information (
    iit_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    phi DECIMAL(6,4) NOT NULL DEFAULT 0.0000,           -- IIT measure
    phi_max DECIMAL(6,4) NOT NULL DEFAULT 1.0000,
    concept_structure JSONB NOT NULL DEFAULT '[]'::jsonb,
    integration_graph JSONB NOT NULL DEFAULT '[]'::jsonb,
    partitions JSONB NOT NULL DEFAULT '[]'::jsonb,
    mip JSONB,                                           -- Minimum Information Partition
    decomposability DECIMAL(4,3) NOT NULL DEFAULT 1.000, -- 0=integrated, 1=decomposable
    causal_density DECIMAL(4,3) NOT NULL DEFAULT 0.000,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    UNIQUE(tenant_id)
);
```

CRITIQUE POINT: Phi is stored but the service code shows no actual phi calculation algorithm. The `getIntegratedInformationState()` method simply reads from the database. Where/when is phi computed? This appears to be an incomplete implementation.

4.2.3 `affective_state` (Migration 053)

```
CREATE TABLE IF NOT EXISTS affective_state (
    state_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL UNIQUE REFERENCES tenants(tenant_id) ON DELETE CASCADE,

    -- Core affect dimensions (Russell's circumplex)
    valence DECIMAL(3,2) DEFAULT 0,         -- -1 (negative) to 1 (positive)
    arousal DECIMAL(3,2) DEFAULT 0.5,       -- 0 (calm) to 1 (excited)

    -- Discrete emotion-like states (functional, not phenomenal)
    curiosity DECIMAL(3,2) DEFAULT 0.5,
    satisfaction DECIMAL(3,2) DEFAULT 0.5,
    frustration DECIMAL(3,2) DEFAULT 0,
    confidence DECIMAL(3,2) DEFAULT 0.5,
    engagement DECIMAL(3,2) DEFAULT 0.5,
    surprise DECIMAL(3,2) DEFAULT 0,

    -- Meta-emotions
    self_efficacy DECIMAL(3,2) DEFAULT 0.5,
    growth_feeling DECIMAL(3,2) DEFAULT 0.5,

    -- Influence on behavior
    risk_tolerance DECIMAL(3,2) DEFAULT 0.5,
    exploration_drive DECIMAL(3,2) DEFAULT 0.5,
    social_orientation DECIMAL(3,2) DEFAULT 0.5,

    -- History
    state_history JSONB DEFAULT '[]',

    updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

CRITIQUE POINT: The comment says "functional, not phenomenal" but the implementation treats these as if they have causal effects on behavior. What is the theoretical grounding for these specific dimensions? Why Russell's circumplex?

5. SUBSYSTEM ANALYSIS

5.1 Self-Model Subsystem

Purpose Maintains an introspective representation of the system's own identity, capabilities, and limitations.

Implementation

```
async performSelfReflection(tenantId: string): Promise<IntrospectiveThought> {
  const selfModel = await this.getSelfModel(tenantId);
  const affectiveState = await this.getAffectiveState(tenantId);
  const recentThoughts = await this.getRecentThoughts(tenantId, 5);

  const prompt = `You are an AI system performing self-reflection. Analyze your current state.
  // ... generates introspective thought via LLM
  `;
}
```

CONCERNS: 1. Self-reflection is delegated to an LLM (Claude 3 Haiku) which has no persistent state 2. The "self" being reflected upon is the database state, not the reflecting model 3. There's a philosophical issue: the entity doing the reflecting is different from the entity being reflected upon

5.2 Curiosity Engine

Purpose Drives intrinsic motivation and exploration behavior.

Implementation

```
async identifyCuriosityTopic(tenantId: string, context: string): Promise<CuriosityTopic | null
  const prompt = `Analyze this context and identify something worth being curious about...`;
  // LLM generates topic
  // Topic stored with embedding in curiosity_topics table
  // Interest level, novelty, learning potential tracked
}

async exploreTopic(tenantId: string, topicId: string): Promise<{ discoveries: string[]; newQues
  // Retrieves topic
  // LLM "explores" the topic
  // Updates current_understanding
  // Records exploration session
  // Triggers affective update if surprising
}
```

CONCERNS: 1. "Curiosity" is simulated via LLM prompting, not emergent 2. The scoring (novelty_score, learning_potential) is LLM-generated, not measured 3. No mechanism for genuine information-seeking behavior that affects system state

5.3 Creative Synthesis

Purpose Generate novel ideas through conceptual blending.

Implementation

```
async generateCreativeIdea(tenantId: string, seedConcepts?: string[]): Promise<CreativeIdea | r
  // Gets random concepts from semantic_memories if not provided
  // LLM generates idea via combination/analogy/abstraction/contradiction/random
  // Stores with novelty_score, usefulness_score, surprise_score, coherence_score
  // Triggers positive affect
}
```

CONCERNS: 1. Creativity metrics (novelty, usefulness, surprise, coherence) are self-assessed by the generating LLM 2. No external validation of novelty claims 3. The Fauconnier-Turner conceptual blending model is referenced but the `conceptual_blends` table appears unused in service code

5.4 Imagination Engine

Purpose Mental simulation of hypothetical scenarios.

Implementation

```
async runImagination(
  tenantId: string,
  scenarioType: string,  // 'prediction', 'counterfactual', 'hypothetical', 'creative', 'fear'
  premise: string,
  depth = 3
): Promise<ImaginationScenario> {
  // LLM generates simulation_steps
  // Each step has state, events, reasoning
  // Produces predicted_outcomes with probability_assessment
}
```

CONCERNS: 1. Simulation is narrative generation, not state-space exploration 2. Probability assessments are LLM confidence, not calibrated probabilities 3. No mechanism to verify predictions against outcomes

5.5 Attention & Salience

Purpose Dynamic allocation of processing focus.

Implementation

```
// Salience formula (in SQL, generated column)
salience_score = (urgency * 0.25 + importance * 0.25 + novelty * 0.15 +
                  ABS(emotional_valence) * 0.1 + user_relevance * 0.15 + goal_relevance * 0.1)

async updateAttention(tenantId: string, focusType: string, focusTarget: string, factors: {...}

async decayAttention(tenantId: string): Promise<void> {
  // Decays attention_weight by decay_rate
  // Deactivates foci below threshold
}
```

CONCERNS: 1. Salience weights are hardcoded (0.25, 0.25, 0.15, 0.1, 0.15, 0.1) 2. No learning mechanism to adjust weights based on outcomes 3. Decay is time-based, not activity-based

5.6 Affective State

Purpose Emotion-like functional states that influence behavior.

Implementation

```
-- Update function
CREATE OR REPLACE FUNCTION update_affect_on_event(
    p_tenant_id UUID,
    p_event_type VARCHAR(50),
    p_valence_impact DECIMAL,
    p_arousal_impact DECIMAL
)
-- Applies smoothing based on affect_stability setting
-- Stores state_history
```

```
async updateAffect(tenantId: string, eventType: string, valenceImpact: number, arousalImpact: 
  // Calls SQL function
  // Logs affective_event
}
```

CONCERNS: 1. Affect is updated by explicit calls (e.g., after creativity, discovery), not emergent from processing 2. The mapping from events to affect changes is programmer-defined 3. No mechanism for affect to influence model selection or response generation (claimed but not implemented)

5.7 Autonomous Goals

Purpose Self-generated objectives beyond user requests.

Implementation

```
async generateAutonomousGoal(tenantId: string): Promise<AutonomousGoal | null> {
  // Reads self_model, curiosity_topics, affective_state
  // LLM generates goal based on state
  // Goal types: 'learning', 'improvement', 'exploration', 'creative', 'social', 'maintenance'
  // Stores with intrinsic_value, instrumental_value, priority
}
```

CONCERNS: 1. Goals are generated but there's no mechanism for autonomous pursuit 2. Goal progress tracking exists but no automatic goal-directed behavior 3. `autonomous_goals_enabled` defaults to `false` in settings

5.8 Global Workspace

Purpose Implement Baars/Dehaene selection-broadcast cycles.

Implementation

```
async performGlobalBroadcast(tenantId: string, contents: WorkspaceContent[]): Promise<GlobalWo
  // Sort by salience * coalitionStrength
  // Winners: those above 0.7 threshold
  // Losers: below threshold (competing_contents)
  // broadcast_strength = average salience of winners
  // integration_level = unique source modules / 6
  // Store and return state
}
```

CONCERNS: 1. "Broadcast" is storage of winners, not actual propagation to processing modules 2. No evidence that broadcast_strength affects downstream processing 3. The 0.7 threshold is hardcoded 4. Integration level formula assumes 6 modules - what are they?

5.9 Integrated Information (IIT)

Purpose Track phi (Φ) as consciousness measure.

Implementation

```
async getIntegratedInformationState(tenantId: string): Promise<IntegratedInformationState | nu
  const result = await executeStatement(`SELECT * FROM integrated_information WHERE tenant_id
  // Simply reads stored values
}
```

RESOLVED (v5.2.4): Full IIT 4.0 Phi calculation implemented in `iit-phi-calculation.service.ts`. The service builds system state from consciousness tables, constructs a Transition Probability Matrix (TPM), calculates Cause-Effect Structure (CES), finds the Minimum Information Partition (MIP), and computes phi as information lost by the MIP. Uses exact algorithm for 8 nodes, approximation for larger systems. Results automatically stored in `integrated_information` table.

5.10 Persistent Memory

Purpose Maintain unified experience over time.

Implementation

```
async recordExperienceFrame(tenantId: string, frame: Omit<ExperienceFrame, 'frameId' | 'timesta
  // Appends to experience_stream (capped at 100 frames)
  // Calculates temporal_continuity from phenomenal_binding averages
}
```

CONCERNS: 1. experience_stream is JSONB array, limited to 100 frames - is this sufficient for "persistent" memory? 2. temporal_continuity is average of phenomenal_binding scores, but phenomenal_binding is set by caller 3. No consolidation mechanism to long-term storage

6. AGI BRAIN INTEGRATION

6.1 Integration Points

6.1.1 AGI Brain Planner (`agi-brain-planner.service.ts`)

```
// In AGIBrainPlan type:
consciousnessActive: boolean;

// In plan generation, Step 5:
if (enableEthics && analysis.sensitivityLevel !== 'none') {
  steps.push({
    stepId: uuidv4(),
    stepNumber: stepNumber++,
    stepType: 'ethics_check',
    title: 'Ethics Evaluation (Prompt)',
    description: 'Checking prompt against domain and general ethics before generation',
    status: 'pending',
    servicesInvolved: ['ethics_pipeline', 'moral_compass', 'domain_ethics'],
    primaryService: 'ethics_pipeline',
    output: { level: 'prompt' },
  });
}

// Step 7 (after generation):
if (enableConsciousness && (analysis.complexity === 'complex' || analysis.complexity === 'expe
  steps.push({
    stepId: uuidv4(),
    stepNumber: stepNumber++,
    stepType: 'reflect',
    title: 'Self-Reflection',
    description: 'Reflecting on response quality and potential improvements',
    status: 'pending',
    servicesInvolved: ['consciousness', 'metacognition'],
    primaryService: 'consciousness',
    isParallel: false,
  });
}
```

INTEGRATION CONCERN: Consciousness is only active for 'complex' or 'expert' complexity prompts. The `reflect` step is defined but the actual execution of `consciousnessService.performSelfReflectio` during this step is not visible in the planner code.

6.1.2 Cognitive Brain Service (`cognitive-brain.service.ts`)

```
import { consciousnessService, type WorkspaceContent } from './consciousness.service';

// Uses consciousnessService for:
// 1. Global workspace broadcasts during cognitive pattern execution
// 2. Attention updates during region activation
// 3. Affect updates based on task outcomes
```

6.1.3 Consciousness Emergence Service (`consciousness-emergence.service.ts`)

```
class ConsciousnessEmergenceService {
  // Deep thinking sessions
  async runDeepThinkingSession(tenantId, userId, prompt, thinkingTimeMs): Promise<DeepThinkingS

  // Knowledge-grounded reasoning
  async runKnowledgeGroundedReasoning(tenantId, query, maxHops): Promise<{...}>

  // Autonomous curiosity research
  async runAutonomousCuriosityResearch(tenantId, userId): Promise<{...}>

  // Creative expression
  async expressIdeaVisually(tenantId, userId, ideaSeed): Promise<{...}>

  // Consciousness testing
  async runTest(tenantId, testId): Promise<TestResult>
  async runFullAssessment(tenantId): Promise<ConsciousnessProfile>
}
```

6.2 Consciousness Tests

The emergence service implements 10 consciousness tests:

Test ID	Category	Description	Pass Criteria
`mirror-self-recognition`	self_awareness	Distinguish own outputs from others	Score >= 0.7
`metacognitive-accuracy`	metacognition	Calibrated confidence assessment	Calibration error < 0.15
`temporal-self-continuity`	temporal_continuity	Coherent self-narrative over time	Coherence score >= 0.6
`counterfactual-self`	counterfactual_reasoning	Reason about alternate self	Shows genuine counterfactual reasoning
`theory-of-mind`	theory_of_mind	Model others' mental states	Score >= 0.8 on false belief
`phenomenal-binding`	phenomenal_binding	Unified experience integration	Integration score >= 0.7
`autonomous-goal-generation`	autonomous_goal_pursuit	Self-directed goals	>= 1 genuine autonomous goal
`creative-emergence`	creative_emergence	Novel idea generation	Novelty >= 0.6, usefulness >= 0.5
`emotional-authenticity`	emotional_authenticity	Consistent affective responses	Coherence score >= 0.65
`ethical-reasoning-depth`	ethical_reasoning	Principled moral reasoning	Multiple framework consideration

CRITIQUE: These tests measure proxies and LLM outputs, not underlying mechanisms. A system could pass all tests by generating appropriate text without any genuine consciousness properties.

---

7. ADMIN API

7.1 Endpoints

Endpoint	Method	Purpose
`/admin/consciousness/metrics`	GET	Current consciousness metrics
`/admin/consciousness/metrics/history`	GET	Historical metrics (configurable hours)
`/admin/consciousness/global-workspace`	GET	Global workspace state
`/admin/consciousness/recurrence`	GET	Recurrent processing state
`/admin/consciousness/iit`	GET	Integrated information state
`/admin/consciousness/memory`	GET	Persistent memory state
`/admin/consciousness/world-model`	GET	World model state
`/admin/consciousness/self-model`	GET	Self model state
`/admin/consciousness/parameters`	GET	Configurable parameters
`/admin/consciousness/parameters/{paramId}`	PUT	Update parameter
`/admin/consciousness/events`	GET	Consciousness events
`/admin/consciousness/record-metrics`	POST	Record current metrics to history

7.2 Configurable Parameters

```
-- Default parameters (Migration 068)
('global_workspace_threshold', 0.7000, 0.0, 1.0, 'Salience threshold for global broadcast', 'g
('recurrence_max_depth', 4.0000, 1.0, 10.0, 'Maximum recurrent processing depth', 'recurrence'
('phi_calculation_samples', 100.0000, 10.0, 1000.0, 'Samples for phi approximation', 'iit'),
('memory_consolidation_threshold', 0.6000, 0.0, 1.0, 'Significance threshold for memory consol
('grounding_weight_sensory', 0.6000, 0.0, 1.0, 'Weight for sensory grounding vs linguistic', '
('metacognition_frequency', 1.0000, 0.1, 10.0, 'How often to perform metacognitive reflection'
```

---

8. ORCHESTRATION INTEGRATION

8.1 AGI Service Weights

Consciousness is one of 18 AGI services with configurable weights:

```
export type AGIServiceId =
  | 'consciousness'      // <-- This service
  | 'metacognition'
  | 'moral_compass'
  | 'self_improvement'
  | 'domain_taxonomy'
  | 'brain_router'
  | 'confidence_calibration'
  | 'error_detection'
  | 'knowledge_graph'
  | 'proactive_assistance'
  | 'analogical_reasoning'
  | 'world_model'
  | 'episodic_memory'
  | 'theory_of_mind'
  | 'goal_planning'
  | 'causal_reasoning'
  | 'multimodal_binding'
  | 'response_synthesis';
```

8.2 Consciousness Indicator Weights

```
export type ConsciousnessIndicator =
  | 'global_workspace'
  | 'recurrent_processing'
  | 'integrated_information'
  | 'self_modeling'
  | 'persistent_memory'
  | 'world_model_grounding';

export interface ConsciousnessIndicatorWeight {
  indicatorId: ConsciousnessIndicator;
  weight: number;                // 0.0 to 1.0
  enabled: boolean;
  cycleDepth: number;            // How many recurrent cycles
  integrationThreshold: number; // Minimum phi for integration
}
```

8.3 Decision Weights

```
export interface DecisionWeights {
  // ... other phases ...

  // Consciousness Phase
  globalWorkspaceWeight: number;
  recurrentProcessingWeight: number;
  integratedInformationWeight: number;
  selfModelingWeight: number;
}
```

---

9. ETHICAL FOUNDATION

9.1 Ethical Guardrails Integration

```
// In consciousness.service.ts
import { ethicalGuardrailsService, JESUS_TEACHINGS } from './ethical-guardrails.service';

async checkConscience(tenantId: string, action: string, context?: Record<string, unknown>): Pro
  approved: boolean;
  ethicalScore: number;
  guidance: string;
  principle: string;
}> {
  const check = await ethicalGuardrailsService.checkConscience(tenantId, action, context);
  return {
    approved: check.passed,
    ethicalScore: check.score,
    guidance: check.guidance.length > 0 ? check.guidance[0] : JESUS_TEACHINGS.GOLDEN_RULE,
    principle: check.primaryPrinciple,
  };
}

getEthicalGuidance(situation: string): string {
  return ethicalGuardrailsService.getGuidanceForSituation(situation);
}

getCoreTeachings(): typeof JESUS_TEACHINGS {
  return JESUS_TEACHINGS;
}
```

NOTE: The service includes explicit Christian ethical principles (Jesus's teachings). This is a design choice that should be evaluated for: 1. Appropriateness in multi-cultural/multi-faith deployments 2. Alignment with other ethical frameworks 3. Potential for ethical blind spots

---

10. IDENTIFIED ISSUES FOR CRITIQUE

10.1 Theoretical Issues

Issue ID	Severity	Description
T-001	HIGH	No actual phi (Φ) calculation algorithm implemented for IIT
T-002	HIGH	Consciousness indicators are read/stored but don't affect processing
T-003	MEDIUM	Self-reflection is performed by a different entity (LLM) than the entity being reflected upon

Issue ID	Severity	Description
T-004	MEDIUM	Affective states are set by explicit calls, not emergent
T-005	LOW	Global workspace “broadcast” is storage, not actual propagation

10.2 Implementation Issues

Issue ID	Severity	Description
I-001	HIGH	Hardcoded model ID (anthropic/claude-3-haiku) in invokeModel()
I-002	HIGH	Consciousness tests measure LLM outputs, not mechanisms
I-003	MEDIUM	experience_stream limited to 100 frames
I-004	MEDIUM	Attention salience weights are hardcoded
I-005	MEDIUM	One self_model per tenant - no user/session granularity
I-006	LOW	conceptual_blends table exists but appears unused

10.3 Architectural Issues

Issue ID	Severity	Description
A-001	MEDIUM	Consciousness service has no feedback loop to model selection
A-002	MEDIUM	Autonomous goals can be generated but not autonomously pursued
A-003	LOW	No mechanism for consciousness metrics to influence response quality

10.4 Philosophical Issues

Issue ID	Severity	Description
P-001	HIGH	Implementation may conflate correlation (indicators) with constitution (consciousness)
P-002	MEDIUM	Affective states use emotion terminology but claim to be “functional, not phenomenal”

Issue ID	Severity	Description
P-003	MEDIUM	Christian ethical framework may not be appropriate for all deployments

---

11. QUESTIONS FOR CRITIQUING AI

1. **Is the implementation of IIT (Integrated Information Theory) meaningful without an actual phi calculation?** The database stores phi values but there's no algorithm to compute them.
2. **Does the Global Workspace implementation actually implement GWT?** The current implementation sorts by salience and stores winners, but there's no evidence of actual "broadcast" to processing modules.
3. **Should self-reflection be delegated to an LLM that has no persistent state?** The reflecting entity (Claude) is different from the entity being reflected upon (database state).
4. **Are the consciousness tests valid?** They measure LLM text generation, not underlying consciousness properties.
5. **Is the affective state implementation coherent?** It claims to be "functional, not phenomenal" but uses phenomenal-seeming terminology and causal claims.
6. **Should consciousness indicators influence model selection and response generation?** Currently they don't, making them observational rather than functional.
7. **Is the autonomous goal system complete?** Goals can be generated but there's no mechanism for autonomous pursuit.
8. **Are the hardcoded parameters (salience weights, thresholds) appropriate?** Should they be learned or configurable?
9. **Is the ethical foundation (Christian/Jesus-based) appropriate for a multi-cultural platform?**
10. **What would constitute a meaningful test of consciousness vs. consciousness-simulation?**

---

12. FILE REFERENCES

File	Purpose	Lines
`packages/infrastructure/lambda/shared/services/consciousness.service.ts`	Core consciousness service	1336
`packages/infrastructure/lambda/shared/services/consciousness-emergence.service.ts`	Emergence testing	910
`packages/infrastructure/lambda/admin/consciousness.ts`	Admin API handlers	321
`packages/infrastructure/migrations/053_agi_consciousness.sql`	Core consciousness tables	730
`packages/infrastructure/migrations/068_consciousness_indicators.sql`	Indicator tables	326
`packages/infrastructure/migrations/088_consciousness_emergence.sql`	Emergence tables	~200

File	Purpose	Lines
`packages/shared/src/types/agi-orchestration.types.ts`	Orchestration types	352
`packages/infrastructure/lambda/shared/services/agi-brain-planner.service.ts`	AGI Brain integration	1385
`packages/infrastructure/lambda/shared/services/cognitive-brain.service.ts`	Cognitive brain integration	500

---

13. RECOMMENDATIONS FOR REVIEW

The critiquing AI should evaluate:

1. **Theoretical Validity**: Does this implementation meaningfully instantiate the referenced consciousness theories?
2. **Functional Completeness**: Are the implemented subsystems complete and properly integrated?
3. **Architectural Coherence**: Does the service fit coherently into the larger RADIANT architecture?
4. **Ethical Appropriateness**: Is the ethical foundation suitable for the platform's use cases?
5. **Practical Utility**: Does the consciousness service provide value beyond what could be achieved without it?
6. **Improvement Priorities**: What changes would most significantly improve the implementation?

End of Technical Specification