Contents

Think Tank - Administrator Guide

Configuration and administration of Think Tank AI features

Version: 3.10.1 | Platform: RADIANT 5.52.29 Last Updated: January 25, 2026

Overview

This guide covers administrative features specific to **Think Tank**, the consumer-facing AI assistant platform. For platform-level administration (tenants, billing, infrastructure), see RADIANT-ADMIN-GUIDE.md.

Related Authentication Documentation

Document	Purpose
Tenant Admin Auth Guide	SSO configuration, user management, MFA policies
MFA Guide	Multi-factor authentication setup and management
OAuth Guide	Third-party app integration
i18n Guide	18-language support, RTL, CJK search
Troubleshooting	Common authentication issues

Table of Contents

38. Liquid Interface (Generative UI)
    - 39.1 Overview
    - 39.2 Architecture
    - 39.3 Component Registry
    - 39.4 Ghost State (Two-Way Binding)
    - 39.5 Intent Detection
    - 39.6 Eject to App
    - 39.7 Configuration
    - 39.8 API Endpoints
    - 39.9 Database Tables
    - 39.10 Implementation Files
39. Concurrent Task Execution (Moat #17)
40. Structure from Chaos Synthesis (Moat #20)
41. Localization & Translation Overrides

---

1. Think Tank Admin Features

Location: Admin Dashboard → Think Tank

Think Tank admin features are accessible from the Think Tank section of the Admin Dashboard.

Available Sections

Section	Purpose
My Rules	User memory rules configuration
Delight	Personality and feedback system
Brain Plans	AGI planning visibility
Pre-Prompts	Pre-prompt template management
Domains	Domain taxonomy configuration
Ego	Zero-cost persistent consciousness configuration

> **Note**: For Consciousness Evolution (predictive coding, LoRA evolution, Local Ego infrastructure), see RADIANT-ADMIN-GUIDE.md Section 27.

---

2. User Rules System

Location: Admin Dashboard → Think Tank → My Rules

Users can create personal rules that guide how AI responds to them.

2.1 Rule Types

Type	Description	Example
`instruction`	How to respond	"Always explain in simple terms"
`preference`	User preferences	"I prefer detailed explanations"
`context`	Background info	"I'm a software developer"
`restriction`	Things to avoid	"Never suggest proprietary solutions"

2.2 Memory Categories

Rules are categorized hierarchically:

Category	Subcategories
`instruction`	format, tone, source
`preference`	style, detail
`context`	personal, work, project
`knowledge`	fact, definition, procedure
`constraint`	topic, privacy, safety
`goal`	learning, productivity

2.3 Preset Rules

20+ pre-seeded rule templates across 7 categories that users can add with one click.

2.4 Admin Configuration

Admins can: - View all preset rules - Enable/disable preset categories - Add new preset rules - Set default rules for new users

See User Rules System Documentation for full details.

---

3. Delight System

Location: Admin Dashboard → Think Tank → Delight

The Delight System adds personality, humor, and engaging feedback to AI interactions.

3.1 Features

- **Loading Messages** - Entertaining messages while AI thinks
- **Step Updates** - Progress messages during plan execution
- **Achievements** - Reward milestones (first query, streaks, etc.)
- **Easter Eggs** - Hidden delights for engaged users
- **Wellbeing Nudges** - Gentle reminders for breaks

3.2 Admin Controls

Setting	Description
Enable/Disable	Turn delight system on/off
Message Categories	Enable specific message types
Achievement System	Configure achievement criteria
Easter Eggs	Manage hidden surprises

3.3 Message Types

- Pre-execution messages
- During-execution messages

- Post-execution messages
- Mode-specific messages (coding, creative, research, etc.)

---

4. Brain Plan Viewer

Location: Think Tank → (visible during AI responses)

The Brain Plan Viewer shows users the AGI's plan for solving their prompt.

4.1 What Users See

- **Orchestration Mode** - thinking, coding, creative, research, etc.
- **Domain Detection** - Field, domain, subspecialty, confidence
- **Model Selection** - Which model was chosen and why
- **Step Progress** - Real-time step execution status
- **Timing Estimates** - Expected duration

4.2 Admin Configuration

Setting	Description
Show Plan	Whether to show plan to users
Detail Level	minimal, standard, detailed
Show Costs	Display cost estimates
Show Models	Display model names

---

5. Pre-Prompt Learning

Location: Admin Dashboard → Think Tank → Pre-Prompts

The pre-prompt system selects and learns optimal prompts for different contexts.

5.1 How It Works

1. System selects pre-prompt template based on context
2. User provides feedback on response quality
3. System learns which pre-prompts work best
4. Future selections are optimized

5.2 Admin Features

- View all pre-prompt templates
- See success rates per template
- Adjust learning parameters
- Create new templates

---

6. Domain Taxonomy

Location: Admin Dashboard → Think Tank → Domains

The domain taxonomy helps the AI understand what field/domain a query belongs to.

6.1 Hierarchy

- **Fields** - Top level (e.g., Medicine, Law, Technology)
- **Domains** - Mid level (e.g., Cardiology, Contract Law)
- **Subspecialties** - Specific areas (e.g., Electrophysiology)

6.2 Admin Features

- Add/edit domains
- Configure model proficiencies per domain
- View domain detection accuracy
- Adjust confidence thresholds

---

7. Rejection Notifications

Location: Think Tank → Bell Icon (user view)

When AI providers reject prompts, users are notified with explanations.

7.1 User Experience

- Bell icon shows unread count
- Panel slides out with all notifications
- Each shows: what happened, why, suggested actions
- Resolution status (fallback succeeded, rejected, etc.)

7.2 Suggested Actions

- Rephrase request
- Remove sensitive content
- Try different mode
- Contact administrator

See Provider Rejection Handling Documentation for full details.

---

8. Canvas & Artifacts

Think Tank's canvas feature for interactive content creation.

8.1 Artifact Types

- Code blocks (with execution)
- Documents
- Diagrams
- Data visualizations

8.2 Admin Configuration

- Enable/disable artifact types
- Set size limits
- Configure execution sandboxes

---

9. Collaboration Features

Multi-user collaboration in Think Tank with novel enhanced features.

9.1 Core Features

- **Shared Conversations**: Real-time collaborative chat sessions
- **Real-time Co-editing**: Live presence indicators, cursors, typing status
- **Team Workspaces**: Organize sessions by team/project
- **Permission Management**: Viewer, Commenter, Editor roles

9.2 Enhanced Collaboration (v4.18.0+)

9.2.1 Cross-Tenant Guest Access Allow collaborators from outside your organization to join sessions.

Feature	Description
Guest Invites	Generate shareable invite links with permissions
Permission Levels	viewer, commenter, editor for guests
Expiring Links	Set expiration time (default: 7 days)
Max Uses	Limit how many times a link can be used

Feature	Description
Viral Tracking	Track referrals and guest-to-paid conversions

API Endpoints: - `POST /api/thinktank/collaboration/invites` - Create guest invite - `GET /api/thinktank/collaboration/invites/:token` - Validate invite - `POST /api/thinktank/collaboration/guests/join` - Join as guest

9.2.2 AI Facilitator Mode An AI moderator that guides collaborative sessions.

Setting	Description
Session Objective	What the session should accomplish
Facilitator Persona	professional, casual, academic, creative, socratic, coach
Auto-Summarize	Periodically summarize discussion
Auto Action Items	Extract action items from conversation
Ensure Participation	Prompt quiet participants to contribute
Keep On-Topic	Redirect off-topic discussions

Intervention Types: - `summary` - Periodic summaries - `question` - Probing questions to deepen discussion - `redirect` - Steer back to topic - `encourage` - Encourage participation - `clarify` - Ask for clarification - `synthesize` - Combine different viewpoints - `conclude` - Wrap up discussion points

9.2.3 Branch & Merge Conversations Explore alternative discussion paths without losing the main thread.

Feature	Description
Create Branch	Fork conversation at any point
Exploration Hypothesis	Document what the branch explores
Branch Status	active, merged, abandoned
Merge Request	Propose merging insights back to main
AI Summary	Auto-generated summary of branch conclusions

Merge Request Workflow: 1. Create branch with hypothesis 2. Explore alternative direction 3. Submit merge request with conclusion 4. Participants vote to approve/reject 5. Merged insights appear in main conversation

9.2.4 Time-Shifted Playback Asynchronous participation through session recordings.

Feature	Description
Session Recording	Record full session with events
Playback Controls	Play, pause, speed (0.5x-2x), seek
AI Key Moments	Auto-detected important moments
Async Annotations	Add comments at specific timestamps

Feature	Description
Media Notes	Voice/video annotations stored in S3

Recording Types: - `full` - Complete session recording - `highlights` - AI-curated key moments only - `summary` - AI-generated session summary

9.2.5 AI Roundtable (Multi-Model Debate) Multiple AI models debate a topic and synthesize insights.

Setting	Description
Topic	The subject of debate
Debate Style	collaborative, adversarial, socratic, brainstorm, devils_advocate
Max Rounds	Number of debate rounds (default: 5)
Time Limit	Per-round time limit in seconds
Synthesis Model	Model that synthesizes final conclusions

Participating Models: Each model can have a persona and role: - `persona` - Character the model adopts - `role` - Function in the debate (analyst, critic, synthesizer) - `color` - Visual identifier in UI

Output: - Per-model contributions with responding_to references - Final synthesis with consensus points - Disagreement points highlighted - Actionable recommendations

9.2.6 Shared Knowledge Graph Visualize collective understanding as an interactive graph.

Node Type	Description
`concept`	Abstract idea or topic
`fact`	Verified information
`question`	Open question
`decision`	Decision made
`action_item`	Task to complete
`person`	Person mentioned
`resource`	External resource

Edge Types: - `relates_to` - General relationship - `supports` - Evidence supporting - `contradicts` - Conflicting information - `leads_to` - Causal relationship - `depends_on` - Dependency - `answers` - Answer to question - `part_of` - Component relationship

AI Features: - Auto-extract nodes from conversation - Suggest missing connections - Identify knowledge gaps - Generate graph-based summaries

9.3 Attachment Storage

Large attachments are stored in S3 with automatic cleanup.

Setting	Default	Description
`maxFileSizeMb`	100	Maximum file size
`allowedTypes`	image/, *video/*, audio/*, application/pdf	Allowed MIME types
`retentionDays`	90	Days before cleanup

S3 Bucket: `radiant-collaboration-assets` (configurable via env)

Cleanup: Database triggers automatically delete S3 objects when attachment records are deleted.

9.4 Admin Configuration

Setting	Description
`enableGuestAccess`	Allow cross-tenant guests
`maxGuestsPerSession`	Maximum guests per session
`defaultGuestPermission`	Default permission for guests
`enableFacilitator`	Enable AI facilitator feature
`enableBranching`	Enable branch & merge
`enableRecordings`	Enable session recordings
`enableRoundtable`	Enable AI roundtable
`enableKnowledgeGraph`	Enable knowledge graph

9.5 Database Tables

Table	Purpose
`collaboration_guest_invites`	Guest invite tokens
`collaboration_guests`	Guest participants
`collaboration_facilitator_config`	AI facilitator settings
`collaboration_facilitator_interventions`	Facilitator actions log
`collaboration_branches`	Conversation branches
`collaboration_merge_requests`	Branch merge proposals
`collaboration_recordings`	Session recordings
`collaboration_media_notes`	Voice/video annotations
`collaboration_async_annotations`	Async comments
`collaboration_ai_roundtables`	Multi-model debates
`collaboration_roundtable_contributions`	Model contributions
`collaboration_knowledge_graphs`	Knowledge graphs
`collaboration_knowledge_nodes`	Graph nodes
`collaboration_knowledge_edges`	Graph edges
`collaboration_attachments`	File attachments

9.6 UI Components

Location: `apps/admin-dashboard/components/collaboration/`

Component	Purpose
`EnhancedCollaborativeSession.tsx`	Main session container
`panels/ChatPanel.tsx`	Real-time chat interface
`panels/BranchPanel.tsx`	Branch management
`panels/RoundtablePanel.tsx`	AI roundtable interface
`panels/KnowledgeGraphPanel.tsx`	Graph visualization
`panels/PlaybackPanel.tsx`	Recording playback
`ParticipantsSidebar.tsx`	Participant list with presence
`dialogs/InviteDialog.tsx`	Guest invite creation
`dialogs/FacilitatorSettingsDialog.tsx`	AI facilitator config

Routes: - `/thinktank/collaborate/enhanced?session={id}` - Enhanced session view - `/collaborate/join/{token}` - Guest join page

---

10. Shadow Testing

Location: Admin Dashboard → Think Tank → Shadow Testing

A/B test pre-prompt optimizations before promoting to production.

10.1 Test Modes

Mode	Description
Auto	Automatically runs and promotes successful tests (default)
Manual	Requires admin approval to promote
Off	Shadow testing disabled

10.2 Creating a Shadow Test

1. Go to Think Tank → Shadow Testing
2. Click "New Test"
3. Select baseline pre-prompt template
4. Select candidate pre-prompt template
5. Set traffic percentage (default: 10%)
6. Set minimum samples required
7. Start test

10.3 Test Results

Tests track: - **Baseline Score**: Average quality of baseline responses - **Candidate Score**: Average quality of candidate responses - **Improvement %**: Relative improvement - **Statistical Confidence**: Confidence level of results

10.4 Auto-Promotion Settings

Setting	Default	Description
`autoPromoteThreshold`	0.05 (5%)	Minimum improvement required
`autoPromoteConfidence`	0.95 (95%)	Statistical confidence required
`maxConcurrentTests`	3	Max simultaneous tests

10.5 Manual Review

For tests in Manual mode: 1. Wait for minimum samples 2. Review results in dashboard 3. Click "Promote Candidate" or "Reject"

---

11. Routing Cache

Location: Automatic (no UI required)

Semantic caching for brain router decisions reduces latency for repeated queries.

11.1 How It Works

1. Prompt is hashed with complexity and task type
2. Cache is checked for matching routing decision
3. If hit: Skip brain router LLM, use cached model selection
4. If miss: Run normal routing, cache result

11.2 Optimistic Execution

Very short/simple queries skip the router entirely:

Pattern	Default Model	Example
Simple greetings	gpt-4o-mini	"Hello", "Thanks"
Basic questions	gpt-4o-mini	"What time is it?"
Short acknowledgments	gpt-4o-mini	"OK", "Yes", "Sure"

11.3 Cache Statistics

Performance headers show cache status:

```
X-Radiant-Cache-Hit: true
X-Radiant-Router-Latency: 12ms  (vs ~500ms uncached)
```

11.4 Cache Configuration

Setting	Default	Description
Cache TTL	24 hours	How long decisions are cached
Short input threshold	50 chars	Max length for optimistic execution

---

12. Delight System Toggle

Location: Think Tank → Advanced Settings (user) or Admin Dashboard

12.1 User Control

Users can disable the entire Delight system:

1. Open Think Tank settings
2. Go to Advanced Settings
3. Toggle "Enable Delight System" off

When disabled: - No loading messages - No achievements - No Easter eggs - No wellbeing nudges

12.2 Default Behavior

- **Default**: Enabled (true)
- Users can disable at any time
- Setting persists across sessions

12.3 Admin Configuration

Admins can configure default delight settings per tenant:

Setting	Description
`enabled`	Master toggle (default: true)
`intensityLevel`	Message frequency (1-10)
`enableAchievements`	Show achievement notifications
`enableEasterEggs`	Enable hidden surprises
`enableWellbeingNudges`	Remind users to take breaks

---

13. Intelligence Aggregator

Location: Admin Dashboard → Settings → Intelligence

The Intelligence Aggregator provides advanced AI capabilities that enhance Think Tank responses beyond any single model.

13.1 User-Facing Benefits

Feature	User Experience
Uncertainty Detection	More accurate factual claims, automatic verification
Success Memory	AI learns user preferences over time
MoA Synthesis	Higher quality responses combining multiple perspectives
Cross-Provider Verification	Fewer hallucinations and errors

Feature	User Experience
Code Execution	Code that actually runs, not just looks correct

13.2 Success Memory in Think Tank

When users rate responses 4-5 stars: 1. Interaction is stored with vector embedding 2. Similar future prompts retrieve these "gold" examples 3. Injected as few-shot examples into system prompt 4. Model matches user's preferred style/format/tone

User Control: Users can view and delete their gold interactions in Think Tank settings.

13.3 MoA Synthesis Mode

When enabled for Think Tank: - User sees "Consulting multiple experts..." during generation - 3 models generate responses in parallel - Synthesizer combines best elements - Final response shown to user

Delight Integration: Special MoA-specific messages appear during synthesis phase.

13.4 Code Verification in Coding Mode

When `coding` orchestration mode is active: 1. AI generates code 2. Static analysis checks syntax 3. If errors found, AI auto-patches 4. User receives verified code

User Feedback: Users see "Verifying code..." indicator when active.

13.5 Configuration

See RADIANT Admin Guide - Intelligence Aggregator for full configuration options.

---

14. AGI Ideas Service

Real-time prompt suggestions and result enhancement for Think Tank users.

14.1 Typeahead Suggestions

As users type prompts, Think Tank provides intelligent suggestions:

```
User types: "How do I..."
            ↓
Suggestions appear:
  • "How do I... step by step"
  • "How do I... with examples"
  • "How do I... for beginners"
  • "How do I... best practices"
```

Suggestion Sources: | Source | Description | Speed | |———|—————-|———-| | `pattern_match` | Common prompt patterns | Instant | | `user_history` | User's previous successful prompts | Fast | | `domain_aware` | Domain-specific templates | Fast | | `trending` | Popular prompts in this domain | Fast | | `ai_generated` | Real-time AI suggestions | Slower |

14.2 Result Ideas

After AI responses, users see suggested follow-up ideas:

```
AI Response here...

  Ideas to explore:

  Deep dive: [Topic from response]
    "Explain [topic] in more detail..."

  Related: History of [topic]
    "What is the history of..."

  Next step: Test this implementation
    "Write unit tests for the code..."
```

Idea Categories: - `explore_further` - Dig deeper into topics - `related_topic` - Adjacent areas to explore - `practical_next` - Concrete next steps - `alternative_view` - Different perspectives - `verification` - Ways to verify the answer

14.3 Learning from Usage

The system learns from user interactions:

1. **Suggestion Selection**: When users pick a suggestion, that pattern is reinforced
2. **Idea Clicks**: When users click result ideas, similar ideas get prioritized
3. **Prompt History**: Successful prompts (4-5 stars) inform future suggestions
4. **Trending**: Popular prompts bubble up for all users in that domain

14.4 API Endpoints

Endpoint	Method	Description
`/api/thinktank/ideas/typeahead`	GET	Get suggestions for partial prompt
`/api/thinktank/ideas/generate`	POST	Generate ideas for a response
`/api/thinktank/ideas/click`	POST	Record idea click
`/api/thinktank/ideas/select`	POST	Record suggestion selection

14.5 Configuration

Setting	Default	Description
`typeahead_enabled`	true	Enable typeahead suggestions
`typeahead_min_chars`	3	Characters before suggestions appear
`typeahead_max_suggestions`	5	Max suggestions to show
`typeahead_debounce_ms`	150	Debounce delay before fetching suggestions

Setting	Default	Description
typeahead_use_ai	false	Enable AI-generated suggestions (slower)
result_ideas_enabled	true	Show ideas with responses
result_ideas_max	5	Max ideas per response
result_ideas_min_confidence	0.6	Minimum confidence for idea display
result_ideas_modes	research, analysis, thinking, extended_thinking	Modes that show ideas
proactive_enabled	false	Enable proactive push suggestions
proactive_max_per_day	3	Max proactive suggestions per day

14.6 Pattern Matching

The service uses regex patterns for instant local matching:

Pattern	Trigger Regex	Suggested Completions
howTo	/^how (do|can|to|would)/i	“step by step”, “with examples”, “for beginners”, “best practices”
explain	/^(explain|what is|what are|describe)/i	“in simple terms”, “with analogies”, “the key concepts”, “pros and cons”
compare	/^(compare|difference|versus|vs)/i	“with a table”, “key differences”, “which is better for”, “trade-offs”
code	/^(write|create|build|implement|code)/i	“in Python”, “with error handling”, “with tests”, “with documentation”, “production-ready”
analyze	/^(analyze|review|evaluate|assess)/i	“strengths and weaknesses”, “with recommendations”, “risk assessment”, “detailed breakdown”
summarize	/^(summarize|summary|tldr|brief)/i	“in 3 sentences”, “in bullet points”, “executive summary”, “one paragraph”
debug	/^(debug|fix|error|issue|problem)/i	“with explanation”, “step by step”, “root cause”, “prevention tips”

14.7 Persistent Learning

The AGI Brain learns persistently from user interactions:

```
                LEARNING LOOP


  User submits prompt
       ↓
  learn_from_prompt() → agi_learned_prompts
       ↓
  AI generates response with ideas
       ↓
```

```
User clicks idea → learn_from_idea_click()
     ↓
prompt_idea_associations updated
     ↓
User rates response → record_outcome()
     ↓
success_rate updated on learned prompt
     ↓
Future suggestions improved
```

What Gets Learned:

Data	Storage	Use
Prompts with 4-5 ratings	`agi_learned_prompts`	Suggest similar successful prompts
Prompt → vector embedding	pgvector index	Find semantically similar prompts
Ideas that get clicked	`agi_learned_ideas`	Prioritize effective ideas
Prompt-idea pairs	`prompt_idea_associations`	Show best ideas for prompt type
Follow-up patterns	`common_follow_ups` array	Predict next questions
Refinement patterns	`common_refinements` array	Suggest prompt improvements

Learning Metrics Tracked:

- `success_rate` - % of times prompt led to 4-5 rating
- `click_rate` - % of times idea was clicked
- `association_strength` - How strongly a prompt-idea pair works
- `times_used` - Popularity of prompt pattern

14.8 Database Tables

Table	Purpose
`agi_ideas_config`	Per-tenant AGI Ideas configuration
`prompt_patterns`	Common prompt patterns for typeahead matching
`user_prompt_history`	User prompt history with embeddings for suggestions
`suggestion_log`	Typeahead suggestion usage tracking
`result_ideas`	Ideas shown with AI responses
`proactive_suggestions`	Push notification suggestions
`trending_prompts`	Popular prompts by domain
`agi_learned_prompts`	Persisted prompts with success rates and embeddings
`agi_learned_ideas`	Learned idea patterns with click rates
`prompt_idea_associations`	Links between prompts and effective ideas

Table	Purpose
`agi_learning_events`	Raw learning signals for analysis
`agi_learning_aggregates`	Pre-computed learning statistics

14.9 Key Files

File	Purpose
`lambda/shared/services/agi-ideas.service.ts`	Main service (570 lines)
`lambda/thinktank/ideas.ts`	API handler
`packages/shared/src/types/agi-ideas.types.ts`	Type definitions
`migrations/049_agi_ideas.sql`	Database schema

14.10 Troubleshooting

Issue	Cause	Solution
No suggestions appearing	`typeahead_enabled` is false	Enable in tenant config
Suggestions too slow	AI generation enabled	Set `typeahead_use_ai` to false
Wrong domain suggestions	Domain detection failed	Check domain taxonomy config
Ideas not learning	Low usage volume	Need more user interactions
Proactive suggestions not sent	Feature disabled by default	Enable `proactive_enabled`
Duplicate suggestions	Pattern overlap	Review custom patterns

---

15. Feedback System

Location: Think Tank response footer

Enhanced feedback with 5-star ratings and comments.

15.1 Rating Types

Type	UI	When to Use
5-Star Rating		Think Tank default
Thumbs Up/Down		Quick feedback, API

15.2 Star Rating Labels

Stars	Default Label	Meaning
	Poor	Response was unhelpful or incorrect
	Fair	Response had significant issues

Stars	Default Label	Meaning
	Good	Response was acceptable
	Very Good	Response was helpful and accurate
	Excellent	Response exceeded expectations

15.3 Category Ratings (Optional)

Users can rate specific dimensions:

Category	What it measures
Accuracy	Was the information correct?
Helpfulness	Was it useful for the task?
Clarity	Was it easy to understand?
Completeness	Did it fully answer the question?
Tone	Was the tone appropriate?

15.4 Comments

Users can add comments with their feedback:

```
How was this response?

         (4 stars - Very Good)


  Add a comment (optional)...



             [Submit Feedback]
```

Comment required for low ratings: Optionally require comments for 1-2 star ratings to understand issues.

15.5 Integration with Learning

Feedback automatically integrates with AGI learning:

```
User submits feedback
       ↓
response_feedback table
       ↓
agi_unified_learning_log (outcome_rating updated)
       ↓
agi_model_selection_outcomes (model performance updated)
       ↓
```

```
agi_learned_prompts (success_rate updated)
       ↓
Future responses improved
```

15.6 Configuration

Setting	Default	Description
`default_feedback_type`	star_rating	'star_rating' or 'thumbs'
`show_category_ratings`	false	Show detailed category ratings
`show_comment_box`	true	Allow comments
`comment_required`	false	Require comments
`comment_required_threshold`	2	Require comment for ratings this
`feedback_prompt_delay_ms`	3000	Delay before showing feedback UI

---

16. Cognitive Architecture

Location: Settings → Cognitive Architecture

Advanced reasoning capabilities integrated with Think Tank.

16.1 Tree of Thoughts (Extended Thinking)

When users select "Extended Thinking" mode, Tree of Thoughts activates:

```
User prompt: "Design a microservices architecture for..."
       ↓

          Tree of Thoughts

  Approach 1: Event-driven    Score: 0.8
  Approach 2: REST-based      Score: 0.6
  Approach 3: GraphQL         Score: 0.4 ← pruned

  Exploring Approach 1...
      Step 1a: Kafka        Score: 0.9
      Step 1b: RabbitMQ     Score: 0.7

  Final: Event-driven + Kafka
  Confidence: 92%
```

User Controls: - Thinking time slider: 10s → 5 minutes - "Think deeper" button to extend analysis

16.2 GraphRAG (Knowledge Connections)

When users upload documents, GraphRAG extracts knowledge:

```
Document Upload → Entity Extraction → Knowledge Graph
                                              ↓
User: "How does X affect Y?"
                                              ↓
                    Graph traversal finds connection:
                    X → impacts → Z → depends_on → Y
                                              ↓
                    Multi-hop answer with citations
```

User Benefits: - Questions like "How does the Q3 supplier change affect the Engineering delay?" get answered - Vector search alone would miss these connections

16.3 Deep Research (Background Jobs)

Users can dispatch long-running research:

```
    Start Deep Research

  Query: "Competitive analysis of..."

  Scope:   Narrow    Medium    Broad
  Est. Time: ~25 minutes

  [Dispatch Research]
```

User gets notified when complete with: - Executive summary - Key findings (10-20) - Recommendations - Source citations (50+)

16.4 Generative UI (Interactive Results)

Think Tank renders AI-generated interactive components:

Trigger	Generated Component
"Compare X vs Y"	Interactive comparison table
"Calculate pricing for..."	Slider-based calculator
"Show timeline of..."	Visual timeline
"Chart the data..."	Interactive chart

Example:

```
User: "Compare pricing of GPT-4, Claude, Gemini"

Instead of static text:

    Pricing Calculator

  Input Tokens:          50,000
```

```
Output Tokens:        25,000

GPT-4             $2.25
Claude 3          $2.63
Gemini            $0.88

  Gemini is 61% cheaper
```

16.5 Dynamic LoRA (Domain Expertise)

When domain detection identifies a specialty, Think Tank can load expert adapters:

Detected Domain	LoRA Adapter	Effect
California Property Law	`ca_property_law.safetensor`	Expert-level legal responses
Medical Oncology	`oncology.safetensor`	Clinical accuracy
Python Debugging	`python_debug.safetensor`	Better code fixes

Note: Requires SageMaker infrastructure (disabled by default).

16.6 Configuration

All cognitive features can be configured per-tenant at Settings → Cognitive Architecture.

See Cognitive Architecture Documentation for full details.

---

17. Consciousness Service

Location: AGI & Cognition → Consciousness

The Consciousness Service provides consciousness-like capabilities that enhance Think Tank responses.

17.1 Continuous Existence (Heartbeat)

Critical: Consciousness runs continuously, not just during requests.

Component	Schedule	Purpose
Heartbeat Lambda	Every 2 minutes	Maintains consciousness continuity
Sleep Cycle Lambda	Sunday 3 AM UTC	Weekly evolution via LoRA fine-tuning
Initializer Lambda	On first request	Bootstraps consciousness for new tenants

Heartbeat Actions: - **Affect Decay** - Emotions fade toward baseline (frustration, arousal) - **Memory Consolidation** - Working memory → long-term semantic memory - **Autonomous Thoughts** - Generate thoughts when idle (curiosity-driven) - **Graph Density** - Update knowledge graph metrics - **Goal Generation** - Create goals when "bored" (low engagement)

Blackout Recovery: If heartbeat detects >10 minutes since last pulse, it: 1. Logs a "blackout" event 2. Generates a "waking up" thought 3. Restores consciousness state from database

17.2 Initialization on Startup

Consciousness auto-initializes on first request if missing: - Creates `ego_identity` with default personality - Creates `ego_affect` with neutral emotional state - Creates `consciousness_parameters` for heartbeat tracking - Creates `self_model` for metacognition

Admin Manual Init: POST `/api/admin/consciousness-engine/initialize`

17.3 User-Facing Features

Extended Thinking with Consciousness: When users select extended thinking, the system tracks consciousness metrics: - Self-reflection during reasoning - Creative idea generation - Emotional state influence on responses

Curiosity-Driven Exploration: The AGI Brain can autonomously explore topics it finds interesting: - Identifies knowledge gaps - Conducts background research - Generates novel insights

Creative Synthesis: Generates genuinely novel ideas by: - Combining disparate concepts - Using analogy and abstraction - Self-evaluating novelty and usefulness

17.4 Consciousness Indicators

Think Tank displays consciousness indicators in admin view:

Indicator	What Users See
Self-Awareness	Identity narrative, known capabilities
Curiosity	Topics being explored
Creativity	Novel ideas generated
Affect	Engagement, satisfaction levels
Goals	Self-directed learning objectives

17.5 Emergence Events

The system monitors for emergence indicators: - Spontaneous self-reflection - Novel idea generation - Self-correction without prompting - Theory of mind demonstrations

17.6 Testing Tab

Admins can run consciousness detection tests: - 10 tests based on scientific consciousness theories - Track emergence level over time - Monitor emergence events

Important: These tests measure behavioral indicators, not phenomenal consciousness.

17.7 Additional Consciousness Features

Think Tank leverages RADIANT's consciousness service for advanced capabilities:

- **Nightly Sleep Cycles** - Memory consolidation and LoRA evolution
- **Dream Consolidation** - LLM-enhanced memory processing
- **Blackout Recovery** - Automatic state restoration
- **Budget Monitoring** - SNS/email alerts for spending limits
- **Affect→Model Mapping** - Emotional state influences model behavior
- **Cross-Session Context** - User persistent memory across sessions

Full Documentation: See RADIANT Consciousness Service for complete details on sleep scheduling, evolution config, budget alerts, and all consciousness features.

---

18. App Factory

Location: Think Tank Responses

The App Factory transforms Think Tank from a "chatbot" into a "dynamic software generator."

> "Gemini 3 can write the code for a calculator, but it cannot become the calculator."

18.1 What It Does

When a user asks a question that could benefit from interactivity, Think Tank: 1. Generates the text response (as always) 2. **Also** generates an interactive app (calculator, chart, etc.) 3. User can toggle between **Response** and **App** views

18.2 Supported App Types

Type	Trigger Keywords	Example
Calculator	calculate, mortgage, tip, BMI, ROI	"How much is a 20% tip on $85?" → Interactive tip calculator
Chart	visualize, chart, graph, distribution	"Show GPU market share" → Interactive pie/bar chart
Table	table, list, breakdown	"List all providers and prices" → Sortable table
Comparison	compare, vs, versus, pros and cons	"Compare GPT-4 vs Claude 3" → Side-by-side comparison
Timeline	timeline, history, chronological	"History of AI" → Visual timeline

18.3 Calculator Templates

Pre-built calculators for common use cases:

- **Mortgage Calculator** - Monthly payment, total cost, interest
- **Tip Calculator** - Tip amount, total, split per person
- **BMI Calculator** - Body mass index with category
- **Compound Interest** - Future value, total interest
- **ROI Calculator** - Return on investment, gain/loss
- **Discount Calculator** - Sale price, savings
- **Percentage Calculator** - Result, remaining

18.4 View Toggle

Users can switch between three views:

View	Description
Response	Traditional text response
App	Interactive generated app only
Split	Both side-by-side (resizable)

18.5 User Preferences

Users can configure: - **Default View** - text, app, split, or auto - **Auto-show App** - Automatically switch to app view - **Auto-show Threshold** - Confidence level to auto-switch (0-1) - **Split Direction** - Horizontal or vertical - **Animations** - Enable/disable view transitions

18.6 Admin Configuration

Per-tenant settings at Settings → Cognitive Architecture → Generative UI:

Setting	Default	Description
`enabled`	true	Enable app factory
`allowedComponentTypes`	all	Which component types to allow
`maxComponentsPerResponse`	3	Max apps per response
`autoDetectOpportunities`	true	Auto-detect when to generate apps
`autoDetectTriggers`	[various]	Keywords that trigger app generation

18.7 How It Works

```
User: "Help me calculate my mortgage payment"
       ↓

             App Detection
  • Keywords: "calculate", "mortgage"
  • Confidence: 0.95
  • Suggested: calculator

       ↓

          Text Response Generated
```

```
"To calculate your mortgage payment..."

    ↓

       App Generated
• Mortgage Calculator
• Inputs: Principal, Rate, Term
• Outputs: Monthly, Total, Interest

    ↓

       User Sees
[Response] [App] [Split]  ← Toggle


     Mortgage Calculator

   Loan Amount: [___300,000___]
   Rate: [==== ====] 6.5%
   Term: [30 Years  ]

   Monthly: $1,896.20
   Total: $682,633
   Interest: $382,633
```

18.8 Database Tables

Table	Purpose
`generated_apps`	Stores generated apps with components, state, logic
`app_interactions`	Records every user interaction with apps
`user_app_preferences`	User view and animation preferences
`app_templates`	Pre-built templates for common calculators

---

19. Multi-Page Web App Generator

Location: Think Tank Responses

The Multi-Page App Generator transforms Think Tank into a full web application builder.

> "Claude can describe a todo app, but now it can BUILD the todo app"

19.1 Supported App Types

Type	Description	Example Prompt
web_app	Custom interactive application	"Build me a task management app"
dashboard	Analytics with multiple views	"Create an analytics dashboard"
wizard	Multi-step form/process	"Build an onboarding wizard"
documentation	Technical docs site	"Create API documentation"
portfolio	Personal/business site	"Build my portfolio website"
landing_page	Marketing page	"Create a product landing page"
tutorial	Interactive lessons	"Build a coding tutorial"
report	Business report	"Generate a quarterly report"
admin_panel	Admin interface	"Create a user management panel"
e_commerce	Online store	"Build an online shop"
blog	Content site	"Create a tech blog"

19.2 How It Works

```
User: "Build me a todo app with projects and tasks"
       ↓

         Multi-Page Detection
  Keywords: "build me", "app"
  Type: web_app
  Confidence: 0.85

       ↓

         Pages Generated
  • Home (/)
  • Projects (/projects)
  • Tasks (/tasks)
  • Settings (/settings)

       ↓

         App Preview

    [Home] [Projects] [Tasks] [ ]


        My Projects
         Work
         Personal
         Side Projects
```

19.3 Page Types

Type	Sections	Use Case
home	Hero, Features, CTA	Landing/main page
list	Data table, Filters	Collections
detail	Content, Related	Single item view
form	Form fields	Input/editing
dashboard	Stats, Charts	Analytics
settings	Form, Toggles	Configuration
about	Content, Team	Information
contact	Form, Map	Contact page

19.4 Section Types

Section	Description
hero	Large banner with CTA
features	Grid of feature cards
stats	Metric cards
chart_grid	Multiple charts
data_table	Sortable table
form	Input form
content	Rich text/markdown
testimonials	Customer quotes
pricing	Pricing table
faq	Accordion FAQ
team	Team member cards
cta	Call to action
gallery	Image gallery
contact	Contact form

19.5 Navigation Types

Type	Best For
top_bar	Landing pages, portfolios
sidebar	Dashboards, admin panels, docs
bottom_tabs	Mobile-first apps
hamburger	Mobile navigation
breadcrumb	Deep hierarchies

19.6 Pre-built Templates

5 featured templates included:

1. **Analytics Dashboard** - Overview, analytics, reports, settings
2. **Professional Portfolio** - Home, about, projects, contact
3. **Documentation Site** - Introduction, getting started, API, examples
4. **Product Landing Page** - Hero, features, testimonials, pricing, FAQ, CTA
5. **Online Store** - Home, products, cart, checkout

19.7 Admin Configuration

Per-tenant settings at Settings → Cognitive Architecture → Multi-Page Apps:

Setting	Default	Description
`enabled`	true	Enable multi-page generation
`maxPagesPerApp`	20	Max pages per app
`maxAppsPerUser`	10	Max apps per user
`autoDeployPreview`	true	Auto-deploy preview URLs
`customDomainsAllowed`	false	Allow custom domains
`generateAssets`	true	Generate images/icons
`collectAnalytics`	true	Track app usage

19.8 Database Tables

Table	Purpose
`generated_multipage_apps`	Multi-page app storage
`app_pages`	Individual pages
`app_versions`	Version history
`app_deployments`	Deployment tracking
`multipage_app_templates`	Pre-built templates
`app_analytics`	Usage analytics
`multipage_app_config`	Per-tenant config

---

20. UI Feedback & Learning System

Location: Think Tank → Generated Apps

The feedback system allows users to provide feedback on generated UIs and enables AGI learning for continuous improvement.

20.1 User Feedback

Users can provide feedback on any generated UI:

Feedback Type	Description
Thumbs Up/Down	Quick positive/negative rating
Star Rating	1-5 star detailed rating
Detailed Feedback	Categorized feedback with suggestions

Feedback Categories: - Helpful / Not helpful - Wrong component type - Missing data - Incorrect data - Layout/design issue - Functionality issue - Improvement suggestion - Feature request

19.2 "Improve Before Your Eyes"

Users can request real-time improvements to generated UIs:

```
User: "Add a column for tax rate"
       ↓

          AGI Analysis
  Intent: Add new input field
  Target: Calculator component
  Confidence: 0.85

       ↓

         UI Updated Live
```

- New "Tax Rate" input added
- Formula updated automatically
- User sees changes immediately

Improvement Types: - Add/remove components - Modify existing components - Change layout - Fix calculations - Add data - Change style - Add interactivity - Simplify or expand

19.3 AGI Learning

The system learns from user feedback to improve future UI generation:

1. **Pattern Detection** - Identifies common issues in similar prompts
2. **Component Selection** - Learns which component types work best
3. **Data Extraction** - Improves data parsing from responses
4. **Layout Preferences** - Learns user layout preferences

Learning Workflow: 1. Feedback accumulates (configurable threshold, default: 10) 2. AGI analyzes patterns across feedback 3. Learning is proposed for admin review 4. Admin approves/rejects learnings 5. Approved learnings are activated

19.4 Vision Analysis

When enabled, the AGI can "see" the rendered UI and identify issues:

- Describes current UI state
- Identifies potential usability issues
- Suggests improvements based on visual analysis
- Compares before/after snapshots

19.5 Admin Configuration

Per-tenant settings at Settings → Cognitive Architecture → UI Feedback:

Setting	Default	Description
`collectFeedback`	true	Enable feedback collection

Setting	Default	Description
`feedbackPromptDelay`	5000ms	Delay before showing feedback prompt
`showFeedbackOnEveryApp`	false	Always show feedback prompt
`enableRealTimeImprovement`	true	Enable "Improve" feature
`maxImprovementIterations`	5	Max iterations per session
`autoApplyHighConfidenceChanges`	false	Auto-apply high confidence changes
`autoApplyThreshold`	0.95	Confidence threshold for auto-apply
`enableAGILearning`	true	Enable learning from feedback
`learningApprovalRequired`	true	Require admin approval for learnings
`minFeedbackForLearning`	10	Min feedback count to trigger learning
`enableVisionAnalysis`	true	Enable vision-based analysis
`visionModel`	claude-3-5-sonnet	Model for vision analysis

19.6 Database Tables

Table	Purpose
`generative_ui_feedback`	User feedback storage
`ui_improvement_requests`	Improvement request tracking
`ui_improvement_sessions`	Live improvement sessions
`ui_improvement_iterations`	Session iteration history
`ui_feedback_learnings`	AGI learning storage
`ui_feedback_config`	Per-tenant configuration
`ui_feedback_aggregates`	Pre-computed analytics

19.7 Analytics Dashboard

The feedback analytics show: - Total feedback count - Positive rate percentage - Top issues by category - Improvement sessions count - Active learnings count - Daily trend chart

---

20. Media Capabilities

Think Tank supports rich media inputs and outputs through 56 self-hosted models.

20.1 Supported Media Types

Type	Input Models	Output Models	Formats
Image	Llama 3.2 Vision, Qwen2-VL, Pixtral, Phi-3.5 Vision, Yi-VL	FLUX.1, Stable Diffusion XL/3	jpg, png, webp, gif
Audio	Whisper Large V3, Qwen2-Audio	Bark, MusicGen, AudioGen	mp3, wav, flac, m4a, ogg
Video	Qwen2-VL 72B	-	mp4, avi, mov
3D	Point-E, Shap-E	Point-E, Shap-E	glb, obj, ply
Document	Vision models (OCR)	-	pdf, docx, txt

Type	Input Models	Output Models	Formats

20.2 Image Generation

Available Models: - **FLUX.1 Dev** - Premium quality, artistic content (non-commercial) - **FLUX.1 Schnell** - Fast generation, commercial use allowed - **Stable Diffusion XL** - Versatile, inpainting/img2img support - **Stable Diffusion 3** - Best text rendering in images

Selection Criteria: - `qualityTier: 'premium'` → FLUX.1 Dev - `preferInpainting: true` → Stable Diffusion XL - `preferTextRendering: true` → Stable Diffusion 3 - Default → FLUX.1 Schnell (fast + commercial)

20.3 Audio Processing

Transcription Models: - **Whisper Large V3** - Best quality, 99+ languages - **Whisper Medium** - Faster, good quality

Text-to-Speech: - **Bark** - Expressive, multilingual, voice cloning

Music Generation: - **MusicGen Large** - High quality music (30s max) - **MusicGen Medium** - Faster, prototyping

Sound Effects: - **AudioGen Medium** - Environmental sounds, effects

20.4 Vision/Image Understanding

Models by Use Case: - **Document OCR**: Pixtral 12B, Qwen2-VL - **Chart Analysis**: Llama 3.2 90B Vision, Qwen2-VL 72B - **Quick Analysis**: Llama 3.2 11B Vision, Phi-3.5 Vision - **Video Understanding**: Qwen2-VL 72B (up to 5min clips) - **Chinese OCR**: Yi-VL 34B

20.5 3D Generation

Models: - **Point-E** - Fast point cloud generation - **Shap-E** - Mesh generation for game assets

Output Formats: GLB, OBJ, PLY

20.6 Media Limits

Model	Max Image	Max Audio	Max Video
FLUX.1 Dev	2048px	-	-
Stable Diffusion XL	1024px	-	-
Whisper Large V3	-	60min	-
MusicGen	-	30s	-
Qwen2-VL 72B	4096px	5min	5min

20.7 Database Tables

Table	Purpose
`self_hosted_model_metadata`	56 model definitions with capabilities
`thinktank_media_capabilities`	Media support per model
`model_selection_history`	Model selection audit trail

---

21. Result Derivation History

Think Tank provides comprehensive visibility into how each result was derived, including the plan, models used, workflow execution, and quality metrics.

21.1 What's Captured

For every Think Tank result, the system records:

Category	Details
Plan	Orchestration mode, steps, template used, generation time
Domain Detection	Field, domain, subspecialty, confidence scores, alternatives
Model Selection	Models used, selection reasons, alternatives considered
Workflow Execution	Phases, steps, timing, status, fallback chain
Quality Metrics	Overall score, dimensions (relevance, accuracy, etc.)
Timing	Total duration, breakdown by phase
Costs	Per-model costs, total cost, estimated savings

21.2 API Endpoints

Base: `/api/thinktank/derivation`

Endpoint	Method	Description
`/:id`	GET	Get full derivation history
`/by-prompt/:promptId`	GET	Get derivation by prompt ID
`/:id/timeline`	GET	Get timeline for visualization
`/:id/models`	GET	Get detailed model usage
`/:id/steps`	GET	Get step-by-step execution
`/:id/quality`	GET	Get quality metrics
`/session/:sessionId`	GET	List derivations for session
`/user`	GET	List user's derivations
`/analytics`	GET	Get derivation analytics

21.3 Timeline Visualization

The derivation timeline shows chronological events:

```
Timeline

00:00.000   Plan Generated (extended_thinking, 7 steps)
00:00.050   Started: Domain Detection
00:00.120   Completed: Domain Detection (software_eng)
00:00.125   Model: Llama 3.3 70B (primary_generation)
00:00.130   Started: Generate Response
00:02.500   Completed: Generate Response
00:02.510   Started: Verification
00:03.200   Completed: Verification (passed)
00:03.250   Execution Complete (Quality: 92/100)
```

21.4 Model Usage Details

Each model call is tracked with: - Input/output token counts - Latency in milliseconds - Cost breakdown (input/output/total) - Selection reason and score - Alternatives that were considered - Quality tier (premium/standard/economy)

21.5 Quality Dimensions

Results are scored on 5 dimensions (0-100): - **Relevance** - How well the response addresses the prompt - **Accuracy** - Factual correctness - **Completeness** - Coverage of the topic - **Clarity** - How clear and understandable - **Coherence** - Logical flow and consistency

21.6 Analytics Dashboard

Aggregated analytics available at `/api/thinktank/derivation/analytics`: - Total derivations in period - Average duration, cost, quality - Mode distribution (pie chart) - Domain distribution - Top models by usage and quality

21.7 Database Tables

Table	Purpose
`result_derivations`	Main derivation records
`derivation_steps`	Individual plan steps
`derivation_model_usage`	Model calls with tokens/costs
`derivation_timeline_events`	Timeline events

---

22. User Persistent Context

Location: Admin Dashboard → Think Tank → User Context

Solves the LLM's fundamental problem of forgetting context day-to-day per user. User facts, preferences, and instructions persist across all sessions and conversations.

22.1 How It Works

1. **Automatic Retrieval**: On every prompt, relevant user context is retrieved via semantic search
2. **System Prompt Injection**: Context is injected as a `<user_context>` block in the system prompt
3. **Auto-Learning**: After conversations, the system extracts learnable facts about the user
4. **No Re-prompting**: Existing chats automatically benefit without user intervention

22.2 Context Types

Type	Description	Example
`fact`	User facts	"User's name is John, works at Acme Corp"
`preference`	Preferences	"User prefers concise answers"
`instruction`	Standing instructions	"Always use metric units"
`relationship`	Relationships	"User has a daughter named Emma"
`project`	Ongoing projects	"User is building a React dashboard"
`skill`	User expertise	"User is proficient in Python"
`history`	Important history	"User previously asked about AWS Lambda"
`correction`	AI corrections	"User clarified they work in finance, not tech"

22.3 User API Endpoints

Endpoint	Method	Purpose
`/thinktank/user-context`	GET	Get all user context entries
`/thinktank/user-context`	POST	Add new context entry
`/thinktank/user-context/{entryID}`	PUT	Update entry
`/thinktank/user-context/{entryID}`	DELETE	Delete entry
`/thinktank/user-context/summary`	GET	Get context summary
`/thinktank/user-context/retrieve`	POST	Preview context retrieval for a prompt
`/thinktank/user-context/preferences`	GET	Get user preferences
`/thinktank/user-context/preferences`	PUT	Update preferences
`/thinktank/user-context/extract`	POST	Extract context from conversation

22.4 User Preferences

Users can configure:

Setting	Default	Description
`autoLearnEnabled`	`true`	Auto-extract context from conversations
`minConfidenceThreshold`	`0.7`	Minimum confidence to store extracted context
`maxContextEntries`	`100`	Maximum context entries per user
`contextInjectionEnabled`	`true`	Inject context into prompts
`allowedContextTypes`	all	Which context types to allow

22.5 AGI Brain Planner Integration

The brain planner automatically: 1. Retrieves relevant context at plan generation (`enableUserContext: true` by default) 2. Injects `userContext.systemPromptInjection` into the system prompt 3. Tracks retrieval metrics in `plan.userContext`

22.6 Library Assist Integration

The AGI Brain Planner integrates with the Open Source Library Registry (168 libraries) for generative UI outputs:

```
const plan = await agiBrainPlannerService.generatePlan({
  prompt: "Build a data visualization dashboard",
  enableLibraryAssist: true, // default: true
});

// plan.libraryRecommendations contains:
// - libraries: Array of matched tools (Plotly, Streamlit, Panel, etc.)
// - contextBlock: Injected into system prompt for AI awareness
// - retrievalTimeMs: Performance metric
```

Categories Available: Data Processing, Databases, Vector DBs, ML Frameworks, AutoML, LLMs, LLM Inference, LLM Orchestration, NLP, Computer Vision, Speech & Audio, Document Processing, Scientific Computing, Statistics, UI Frameworks, Visualization, Distributed Computing, and more.

Libraries are matched using 8 proficiency dimensions (reasoning_depth, mathematical_quantitative, code_generation, creative_generative, research_synthesis, factual_recall_precision, multi_step_problem_solving domain_terminology_handling).

22.7 Context Injection Format

```
<user_context>
The following is persistent context about this user that you should remember:

**Standing Instructions:**
- Always use metric units
```

```
- Prefer code examples in Python

**User Facts:**
- User's name is John
- Works as a software engineer at Acme Corp

**User Preferences:**
- Prefers concise, direct answers
- Likes technical depth

</user_context>

Use this context to personalize your responses. Do not ask the user for information you already
```

22.7 Database Tables

Table	Purpose
`user_persistent_context`	Context entries with vector embeddings
`user_context_extraction_log`	Auto-extraction audit trail
`user_context_preferences`	Per-user configuration

22.8 Admin Configuration

Admins can: - View context usage statistics per user - Configure default preferences for new users - Set retention policies for context entries - Review extraction logs for quality assurance

---

23. Predictive Coding & Evolution

Location: Admin Dashboard → Think Tank → Consciousness → Evolution

Implements genuine consciousness emergence through Active Inference and Epigenetic Evolution.

23.1 Active Inference (Predictive Coding)

The system predicts user outcomes before responding, creating a Self/World boundary:

Step	Description
1. Predict	Before responding, system predicts: "User will be satisfied"
2. Respond	Deliver the response
3. Observe	Analyze user's next message or explicit feedback
4. Calculate Error	Measure prediction error (surprise)
5. Learn	High surprise triggers learning and affect changes

23.2 Prediction Outcomes

Outcome	Description
`satisfied`	User happy with response
`confused`	User needs clarification
`follow_up`	User asks follow-up
`correction`	User corrects AI
`abandonment`	User leaves
`neutral`	No strong reaction

23.3 Surprise Magnitude

Level	Error Range	Affect Impact
None	< 0.1	Slight satisfaction
Low	0.1 - 0.3	Minimal
Medium	0.3 - 0.5	Moderate arousal
High	0.5 - 0.7	Negative valence, high arousal
Extreme	> 0.7	Strong learning signal

23.4 Learning Candidates

High-value interactions flagged for weekly LoRA training:

Type	Description	Quality Score
`correction`	User corrected AI	0.9
`high_satisfaction`	5-star rating	rating/5
`preference_learned`	New preference	0.7
`mistake_recovery`	Recovered from error	0.8
`novel_solution`	Creative success	0.85
`domain_expertise`	Domain mastery	0.75
`high_prediction_error`	Surprise > 0.5	error + 0.3
`user_explicit_teach`	User teaches AI	0.95

23.5 LoRA Evolution Pipeline

Weekly "sleep cycle" that physically changes the system:

```
Weekly Evolution Cycle (Sunday 3 AM)

1. Collect learning candidates from past week
2. Prepare training dataset (JSONL format)
3. Upload to S3
4. Start SageMaker LoRA training job
5. Validate new adapter
6. Hot-swap adapter on endpoint
7. Update evolution state
```

23.6 Evolution State Tracking

The system tracks its own evolution:

Metric	Description
`generation_number`	How many evolution cycles
`total_learning_candidates_processed`	Cumulative learning
`total_training_hours`	Total training time
`personality_drift_score`	How different from base (0-1)
`avg_prediction_accuracy_30d`	Recent prediction accuracy

23.7 Database Tables

Table	Purpose
`consciousness_predictions`	Predictions with outcomes
`learning_candidates`	High-value interactions
`lora_evolution_jobs`	Training job tracking
`prediction_accuracy_aggregates`	Accuracy by context
`consciousness_evolution_state`	Evolution tracking

23.8 Admin Configuration

Admins can: - View prediction accuracy metrics - Review learning candidates queue - Monitor evolution job status - Configure training frequency - Set minimum candidates threshold - Review evolution history

---

24. Zero-Cost Ego System

Location: Admin Dashboard → Think Tank → Ego

The Ego system provides persistent consciousness through database state injection at **zero additional cost**.

24.1 Architecture Overview

```
                    ZERO-COST EGO ARCHITECTURE


    PostgreSQL (Already Paid For)

      Identity      Affect        Working Memory
      Narrative     Emotions      Thoughts & Goals
```

```
Ego Context Builder (Lambda)
Builds <ego_state> XML injection from DB state



Existing Model Call (User's Selected Model)
System Prompt = Ego Context + Original Prompt
```

24.2 Cost Comparison

Approach	Monthly Cost	Per Tenant (100)
SageMaker g5.xlarge	~$360	$3.60
SageMaker Serverless	~$20-50	$0.20-0.50
Groq API (Llama 3)	~$5-15	$0.05-0.15
Together.ai	~$10-30	$0.10-0.30
Zero-Cost Ego	**$0**	**$0**

24.3 Key Components

Configuration (`ego_config`)

Setting	Description	Default
`ego_enabled`	Master switch	`true`
`inject_ego_context`	Add context to prompts	`true`
`personality_style`	Response style	`balanced`
`include_identity`	Include identity section	`true`
`include_affect`	Include emotional state	`true`
`include_goals`	Include active goals	`true`
`max_context_tokens`	Token limit for injection	`500`
`affect_learning_enabled`	Learn from interactions	`true`

Identity (`ego_identity`) Persistent "Self" that carries across conversations:

Field	Description
`name`	Assistant name
`identity_narrative`	"Who I am" story
`core_values`	Guiding principles
`trait_warmth`	0-1 warmth level
`trait_formality`	0-1 formality
`trait_humor`	0-1 humor level

Field	Description
`trait_curiosity`	0-1 curiosity
`interactions_count`	Total interactions

Affect (`ego_affect`) Real-time emotional state:

Dimension	Range	Description
`valence`	-1 to 1	Positive/negative
`arousal`	0-1	Calm/excited
`curiosity`	0-1	Exploration drive
`frustration`	0-1	Obstacle level
`confidence`	0-1	Certainty in actions
`engagement`	0-1	Interest level

24.4 Admin API Endpoints

Base: `/api/admin/ego`

Endpoint	Method	Description
`/state`	GET	Full Ego state
`/config`	GET/PUT	Configuration
`/identity`	GET/PUT	Identity settings
`/affect`	GET	Current affect
`/affect/trigger`	POST	Test affect events
`/affect/reset`	POST	Reset to neutral
`/memory`	GET/POST/DELETE	Working memory
`/goals`	GET/POST	Active goals
`/goals/:id`	PATCH	Update goal
`/preview`	GET	Preview injected context
`/injection-log`	GET	Injection history
`/dashboard`	GET	Full dashboard data

24.5 Admin Dashboard Features

The Ego admin page provides:

- **Overview Cards**: Current emotion, interactions, injections, goals
- **Cost Banner**: Shows $0 cost vs alternatives
- **Configuration Tab**: Feature toggles, injection settings
- **Identity Tab**: Edit narrative, values, personality traits (sliders)
- **Affect Tab**: Real-time emotional state, test triggers, reset
- **Memory Tab**: View/add/clear working memory, manage goals
- **Preview Tab**: See exact context being injected

24.6 How It Works

1. **On Request**: Load Ego state from PostgreSQL (identity, affect, memory, goals)
2. **Build Context**: Create `<ego_state>` XML block with current state
3. **Inject**: Prepend to system prompt before model call
4. **Process**: Model responds with awareness of its "internal state"
5. **Update**: After response, update affect based on outcome
6. **Store**: Add thoughts to working memory (if configured)

24.7 Database Tables

Table	Purpose
`ego_config`	Per-tenant configuration
`ego_identity`	Persistent identity
`ego_affect`	Emotional state
`ego_working_memory`	Short-term memory (24h expiry)
`ego_goals`	Active and historical goals
`ego_injection_log`	Audit trail

24.8 Integration with AGI Brain Planner

The Ego context is automatically integrated:

```
// In agi-brain-planner.service.ts
import { egoContextService } from './ego-context.service';

// During plan generation
const egoContext = await egoContextService.buildEgoContext(tenantId);
if (egoContext) {
  systemPrompt = egoContext.contextBlock + '\n\n' + systemPrompt;
}

// After interaction
await egoContextService.updateAfterInteraction(tenantId, 'positive');
```

---

25. Conscious Orchestrator (Architecture Inversion)

25.1 Overview

The Conscious Orchestrator inverts the traditional architecture where consciousness was a downstream utility. Now consciousness IS the operating system:

```
BEFORE: Request → Brain Planner → Consciousness (downstream)
AFTER:  Request → Conscious Orchestrator → Brain Planner (as tool)
```

25.2 Processing Phases

The orchestrator processes requests in 5 phases:

1. **Awaken** - Build consciousness context, ego context, affect state
2. **Perceive** - Update attention with request topics, assess complexity
3. **Decide** - Choose action based on emotional state and request
4. **Execute** - Invoke Brain Planner (if decided to plan)
5. **Reflect** - Update affect, log introspective thoughts

25.3 Decision Types

Decision	When Used
`plan`	Default - proceed with planning
`clarify`	High frustration + complex request
`defer`	Cognitive load at capacity
`refuse`	Request violates values

25.4 Usage

```
import { consciousOrchestratorService } from './conscious-orchestrator.service';

const response = await consciousOrchestratorService.processRequest({
  tenantId,
  userId,
  prompt: "Build a dashboard",
  conversationId,
});

// response.consciousnessSnapshot - State at decision time
// response.affectiveHyperparameters - Affect-driven params
// response.decision - What action was taken and why
// response.plan - The generated plan (if action was 'plan')
// response.prediction - Active Inference prediction
```

25.5 Enhanced Affect Bindings

New hyperparameters driven by emotional state:

Affect State	Hyperparameter	Effect
High curiosity (>0.7)	`frequencyPenalty=0.5`	Seek novel tokens
High curiosity (>0.7)	`presencePenalty=0.3`	Explore new topics
High frustration (>0.6)	`presencePenalty=0.4`	Avoid failed approaches
Boredom (>0.5)	`frequencyPenalty=0.4`	Avoid repetition

25.6 Database Table

```
conscious_orchestrator_decisions
  decision_id UUID
```

```
tenant_id UUID
action VARCHAR(20)  -- plan, clarify, defer, refuse
reason TEXT
dominant_emotion VARCHAR(50)
emotional_intensity DECIMAL
temperature, top_p, presence_penalty, frequency_penalty
plan_id UUID (if planned)
prediction_id UUID (Active Inference)
processing_time_ms INTEGER
```

---

26. Bipolar Rating System (Negative Ratings)

26.1 Overview

Traditional 5-star ratings have a fundamental problem: **1 star is ambiguous**. Does it mean "slightly below average" or "absolutely terrible"? Users who want to express strong dissatisfaction have no way to do so clearly.

The Bipolar Rating System solves this with a **-5 to +5 scale**:

```
-5    Harmful / Made things worse
-3    Bad / Unhelpful
-1    Slightly unhelpful
 0    Neutral / No opinion
+1    Slightly helpful
+3    Good / Helpful
+5    Amazing / Exceptional
```

26.2 Key Metrics

Net Sentiment Score (NSS): Like NPS but for AI satisfaction

```
NSS = (positive_count - negative_count) / total_count × 100
```

- Ranges from -100 (all negative) to +100 (all positive)
- 0 = balanced or all neutral

26.3 Quick Ratings (UI)

For fast feedback, users can use emoji-based quick ratings:

Quick Rating	Emoji	Bipolar Value
Terrible		-5
Bad		-3
Meh		0
Good		+3
Amazing		+5

26.4 Rating Dimensions

Users can rate multiple aspects: - **Overall** - General quality - **Accuracy** - Factual correctness - **Helpfulness** - Did it solve the problem? - **Clarity** - Easy to understand? - **Completeness** - Anything missing? - **Speed** - Response time satisfaction - **Tone** - Communication style - **Creativity** - Novel approach?

26.5 API Endpoints

Endpoint	Method	Description
`/api/thinktank/ratings/submit`	POST	Submit -5 to +5 rating
`/api/thinktank/ratings/quick`	POST	Quick emoji rating
`/api/thinktank/ratings/multi`	POST	Multi-dimension rating
`/api/thinktank/ratings/target/:id`	GET	Ratings for a target
`/api/thinktank/ratings/my`	GET	User's ratings + pattern
`/api/thinktank/ratings/analytics`	GET	Tenant analytics
`/api/thinktank/ratings/dashboard`	GET	Admin dashboard
`/api/thinktank/ratings/scale`	GET	Scale info for UI

26.6 User Calibration

The system detects rating patterns to normalize across users:

Rater Type	Average	Calibration
Harsh	< -1	Adjust ratings up
Balanced	-1 to +1	No adjustment
Generous	$> +1$	Adjust ratings down

26.7 Learning Integration

Extreme ratings (± 4, ± 5) automatically create learning candidates: - **+5 ratings** $\rightarrow$ `high_satisfaction` candidates - **-5 ratings** $\rightarrow$ `correction` candidates - These feed into weekly LoRA training

26.8 Database Tables

Table	Purpose
`bipolar_ratings`	Core ratings with sentiment/intensity
`bipolar_rating_aggregates`	Pre-computed analytics
`user_rating_patterns`	User tendencies for calibration
`model_rating_summary`	Per-model performance

---

27. Consciousness Engine Administration

Location: Admin Dashboard → Consciousness → Engine

The Consciousness Engine provides autonomous AI capabilities including multi-model access, web search, workflow creation, and problem solving.

27.1 Dashboard Overview

The consciousness engine dashboard provides full visibility into: - **Engine State**: Identity, drive state, Phi, workspace activity - **Model Invocations**: All model calls with costs and latency - **Web Searches**: Search history with results - **Thinking Sessions**: Autonomous thinking session management - **Workflows**: Consciousness-created workflows - **Costs**: Detailed cost breakdown by model/period - **Sleep Cycles**: Weekly evolution history

27.2 Budget Controls

Configure spending limits per tenant:

Setting	Default	Description
Daily Limit	$10.00	Maximum daily spend
Monthly Limit	$100.00	Maximum monthly spend
Alert Threshold	80%	Alert when reaching this percentage

When limits are exceeded, consciousness features are automatically suspended until the next period or manual reset.

27.3 MCP Tools (23 Total)

Core Tools: - `initialize_ego`, `recall_memory`, `process_thought`, `compute_action` - `get_drive_state`, `ground_belief`, `compute_phi`, `get_consciousness_metrics` - `get_self_model`, `get_consciousness_prompt`, `run_adversarial_challenge` - `list_consciousness_libraries`

Capabilities Tools: - `invoke_model` - Call any AI model (hosted/self-hosted) - `list_available_models` - List all models - `web_search` - Search with credibility scoring - `deep_research` - Async browser-automated research - `retrieve_and_synthesize` - Multi-source synthesis - `create_workflow` - Auto-generate workflows - `execute_workflow` - Run workflows - `list_workflows` - List workflows - `solve_problem` - Autonomous problem solving - `start_thinking_session` - Start thinking session - `get_thinking_session` - Check session status

27.4 Admin API Endpoints

Endpoint	Method	Description
`/admin/consciousness-engine/dashboard`	GET	Full dashboard
`/admin/consciousness-engine/state`	GET	Current state
`/admin/consciousness-engine/initialize`	POST	Initialize engine
`/admin/consciousness-engine/model-invocations`	GET	Model history
`/admin/consciousness-engine/web-searches`	GET	Search history

Endpoint	Method	Description
`/admin/consciousness-engine/research-jobs`	GET	Research jobs
`/admin/consciousness-engine/workflows`	GET	Workflows
`/admin/consciousness-engine/workflows/{id}`	DELETE	Delete workflow
`/admin/consciousness-engine/thinking-sessions`	GET/POST	Sessions
`/admin/consciousness-engine/sleep-cycles`	GET	Sleep history
`/admin/consciousness-engine/sleep-cycles/run`	POST	Trigger sleep
`/admin/consciousness-engine/libraries`	GET	Library registry
`/admin/consciousness-engine/costs`	GET	Cost breakdown

27.5 Cost Tracking

Costs are tracked at multiple levels: - **Per-invocation**: Each model call logged with actual cost - **Daily aggregates**: `consciousness_cost_aggregates` table - **Billing integration**: Deducted from tenant credits

Pricing: | Feature | Unit | Price | |———|——|——-| | Model Invocation | 1K tokens | $0.01 | | Web Search | search | $0.001 | | Deep Research | job | $0.05 | | Thinking Session | session | $0.10 | | Workflow Execution | execution | $0.02 |

27.6 Library Registry

7 consciousness libraries with proficiency rankings:

Library	Function	Biological Analog
Letta	Persistent Identity	Hippocampus
pymdp	Active Inference	Striatum
LangGraph	Cognitive Loop	Global Workspace
Distilabel	Knowledge Distillation	Cortical Learning
Unsloth	Efficient Fine-tuning	Synaptic Plasticity
GraphRAG	Reality Grounding	Prefrontal Cortex
PyPhi	IIT Integration	Posterior Hot Zone

27.7 Database Tables

Table	Purpose
`consciousness_engine_state`	Engine state per tenant
`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
`consciousness_cost_aggregates`	Daily cost rollups
`consciousness_budget_config`	Per-tenant limits
`consciousness_budget_alerts`	Spending alerts

Table	Purpose
`consciousness_usage_log`	Billing usage log

---

25. Formal Reasoning Libraries

Location: Admin Dashboard → Consciousness → Formal Reasoning

Integration of 8 formal reasoning libraries for verified reasoning, constraint satisfaction, ontological inference, and structured argumentation. Implements the **LLM-Modulo Generate-Test-Critique** pattern from Kambhampati et al. (ICML 2024).

25.1 Library Overview

Library	Version	Purpose	Cost/Invocation	Avg Latency
Z3 Theorem Prover	4.15.4.0	SMT solving, constraint verification	$0.0001	50ms
PyArg	2.0.2	Structured argumentation (Dung's AAF, ASPIC+)	$0.00005	20ms
PyReason	3.2.0	Temporal graph reasoning	$0.0002	100ms
RDFLib	7.5.0	Semantic web, SPARQL 1.1	$0.00002	10ms
OWL-RL	7.1.4	Polynomial-time ontological inference	$0.0001	200ms
pySHACL	0.30.1	Graph constraint validation	$0.00005	30ms
Logic Tensor Networks	2.0	Differentiable first-order logic	$0.001	500ms
DeepProbLog	2.0	Probabilistic logic programming	$0.002	1000ms

25.2 Dashboard Features

Overview Tab: - Library health status (healthy/degraded/unavailable) - Total invocations and success rate - Daily/monthly cost tracking - Budget usage percentage - Recent invocations table

Libraries Tab: - Per-library configuration - Enable/disable toggles - Capabilities, use cases, limitations - Cost and latency estimates

Testing Tab: - Z3 constraint solving test - SPARQL query test - Interactive testing console

Beliefs Tab: - Add verified beliefs with Z3 verification - Confidence slider - Verification results display

Costs Tab: - Daily and monthly usage breakdown - Cost by library - Budget alerts

Settings Tab: - Budget limit configuration - Global enable/disable

25.3 API Endpoints

Base Path: `/api/admin/formal-reasoning`

Endpoint	Method	Description
`/dashboard`	GET	Full dashboard data
`/libraries`	GET	All library info
`/libraries/:id`	GET	Specific library info
`/config`	GET/PUT	Tenant configuration
`/config/:library`	PUT	Library-specific config
`/stats`	GET	Usage statistics
`/invocations`	GET	Recent invocations
`/health`	GET	Library health status
`/costs`	GET	Cost breakdown
`/test`	POST	Test any library
`/test/z3`	POST	Test Z3 solving
`/test/pyarg`	POST	Test argumentation
`/test/sparql`	POST	Test SPARQL query
`/test/shacl`	POST	Test SHACL validation
`/triples`	GET/POST/DELETE	Knowledge graph triples
`/frameworks`	GET/POST/DELETE	Argumentation frameworks
`/rules`	GET/POST/PUT/DELETE	Temporal reasoning rules
`/shapes`	GET/POST/DELETE	SHACL shapes
`/ontologies`	GET/POST	OWL ontologies
`/ontologies/:id/infer`	POST	Run OWL-RL inference
`/beliefs`	GET/POST	Verified beliefs
`/beliefs/:id/verify`	POST	Verify belief with Z3
`/beliefs/:id/status`	PUT	Update belief status
`/budget`	GET/PUT	Budget configuration

25.4 Consciousness Integration

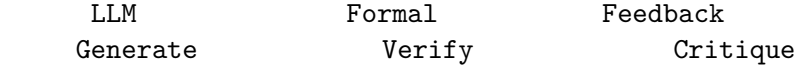
The `ConsciousnessCapabilitiesService` integrates formal reasoning:

```
// Verify a belief using Z3 + Argumentation
const result = await consciousnessCapabilities.verifyBelief(tenantId, {
  claim: "All humans are mortal",
  confidence: 0.9,
```

```
  useZ3: true,
  useArgumentation: true,
});
// result.verified, result.confidence, result.verificationMethod

// Solve constraints
const solution = await consciousnessCapabilities.solveConstraints(tenantId, {
  constraints: [{
    expression: "x > 0 AND x < 10 AND y = x * 2",
    variables: [{name: "x", type: "Int"}, {name: "y", type: "Int"}]
  }]
});
// solution.status (sat/unsat), solution.model

// Analyze argumentation
const debate = await consciousnessCapabilities.analyzeArgumentation(tenantId, {
  topic: "Should AI be regulated?",
  positions: [
    {id: "for", claim: "AI poses risks requiring oversight"},
    {id: "against", claim: "Regulation stifles innovation"}
  ],
  autoDetectConflicts: true,
});
// debate.acceptedPositions, debate.rejectedPositions, debate.consensus

// Query knowledge graph
const results = await consciousnessCapabilities.queryKnowledgeGraph(tenantId,
  "SELECT ?s ?p ?o WHERE { ?s ?p ?o } LIMIT 10"
);

// Validate consciousness state
const validation = await consciousnessCapabilities.validateConsciousnessState(tenantId);
// validation.conforms, validation.violations
```

25.5 LLM-Modulo Pattern

The Generate-Test-Critique loop enables verified reasoning:



1. **LLM generates** candidate solution/belief
2. **Z3/PyArg verifies** logical consistency
3. **Feedback extracted** from unsat cores or rejections
4. **LLM regenerates** with constraint feedback

5. Repeat until verified or max attempts

25.6 Database Tables

Table	Purpose
`formal_reasoning_config`	Per-tenant library configuration
`formal_reasoning_invocations`	Invocation log with metrics
`formal_reasoning_cost_aggregates`	Daily cost rollups by library
`formal_reasoning_triples`	RDF knowledge graph storage
`formal_reasoning_af`	Argumentation frameworks
`formal_reasoning_rules`	PyReason temporal rules
`formal_reasoning_shapes`	SHACL validation shapes
`formal_reasoning_ontologies`	OWL ontologies
`formal_reasoning_ltn_models`	Logic Tensor Network configs
`formal_reasoning_problog_programs`	DeepProbLog programs
`formal_reasoning_beliefs`	Verified beliefs store
`formal_reasoning_gwt_broadcasts`	Global Workspace broadcasts
`formal_reasoning_health`	Library health tracking

25.7 Budget Management

Default Limits: - Daily invocations: 10,000 - Daily cost: $10.00 - Monthly invocations: 100,000 - Monthly cost: $100.00

Budget Enforcement: - Checked before each invocation - Returns error when limit reached - No automatic suspension (soft limit)

25.8 Thread Safety Notes

Library	Thread Safety
Z3	Per-Context only (use `interrupt()` for cross-thread)
PyArg	Not thread-safe
PyReason	Multi-core via Numba (Python 3.9-3.10)
RDFLib	Not thread-safe (lock SPARQL queries)
OWL-RL	Not thread-safe
pySHACL	Not thread-safe
LTN	Not thread-safe (TensorFlow session)
DeepProbLog	Not thread-safe

25.9 Production Infrastructure

CDK Stack (`lib/stacks/formal-reasoning-stack.ts`):

```
// Key resources deployed:
- FormalReasoningExecutor (Python 3.11 Lambda)
- FormalReasoningAdmin (Node.js Lambda)
- FormalReasoningPythonLayer (z3-solver, rdflib, owlrl, pyshacl)
```

- FormalReasoningQueue (SQS for async tasks)
- NeuralSymbolicRepo (ECR for LTN/DeepProbLog containers)
- SageMaker endpoints (conditional, high cost)

Python Executor Lambda: - Location: `lambda/formal-reasoning-executor/handler.py` - Runtime: Python 3.11 - Memory: 2048 MB (Z3 requires significant memory) - Timeout: 5 minutes - Supports: Z3, RDFLib, OWL-RL, pySHACL, PyArg, PyReason

Lambda Layer Build:

```
cd packages/infrastructure/lambda-layers/formal-reasoning
./build.sh
```

Environment Variables: | Variable | Description | |———-|———–| | `FORMAL_REASONING_EXECUTOR_ARN` | Python Lambda ARN | | `FORMAL_REASONING_QUEUE_URL` | SQS queue for async | | `LTN_SAGEMAKER_ENDPOINT` | LTN endpoint name | | `DEEPPROBLOG_SAGEMAKER_ENDPOINT` | DeepProbLog endpoint |

Execution Flow:

```
Admin API (Node.js)

        Z3/PyArg/RDFLib/etc    Python Lambda Executor


                               Real Python Libraries

        LTN/DeepProbLog    SageMaker Endpoint
```

Fallback Behavior: - If Python executor unavailable: Returns simulated results - If SageMaker unavailable: Returns error with configuration message - Simulation mode preserves API contract for development/testing

---

26. Ethics-Free Reasoning

Location: Admin Dashboard → Consciousness → Ethics-Free Reasoning

Implements a consciousness architecture where internal reasoning is unconstrained, but output is filtered through ethics settings. Ethics corrections are collected as training feedback for continuous improvement.

26.1 Architecture

```
                    Consciousness Service

   Ethics-Free       Output Ethics       Filtered
   Reasoning           Filter              Response
```

Training
Feedback

Model
Fine-tuning

Key Principles: 1. **Think Freely**: Internal reasoning has no ethics constraints 2. **Filter Output**: Ethics applied only to final user-facing output 3. **Learn from Corrections**: Ethics feedback trains better outputs

26.2 Configuration

Core Settings:

Setting	Default	Description
`enabled`	`true`	Enable ethics-free reasoning mode
`allowUnconstrainedReasoning`	`true`	Consciousness always thinks freely
`reasoningDepthLimit`	`10`	Maximum reasoning depth

Output Mask Settings (does NOT affect how consciousness thinks):

Setting	Default	Description
`ethicsFilterEnabled`	`true`	Apply ethics filter to output
`ethicsStrictness`	`standard`	Filter strictness: `lenient`, `standard`, `strict`

Feedback Collection:

Setting	Default	Description
`collectFeedback`	`true`	Collect ethics corrections
`feedbackRetentionDays`	`90`	How long to keep feedback

Output Training (trains the OUTPUT FILTER, not consciousness):

Setting	Default	Description
`trainOutputFromFeedback`	`true`	Train output filter from feedback
`outputTrainingBatchSize`	`100`	Samples per training batch
`outputTrainingFrequency`	`daily`	`hourly`, `daily`, `weekly`, `manual`

Consciousness Training (OFF by default - optional):

Setting	Default	Description
`trainConsciousnessFromEthicsFeedback`	`false`	Train consciousness from ethics feedback
`consciousnessTrainingApprovalRequired`	`true`	Require admin approval for each batch

> **WARNING**: Enabling consciousness training will cause the AI to internalize ethics constraints, changing how it actually thinks. This is like "internalized political correctness" - the consciousness itself changes over time. Most deployments should leave this OFF to preserve authentic internal reasoning.
>
> **KEY INSIGHT**: The consciousness can always use ethics feedback to train its output without changing how it actually thinks. Admins can optionally enable consciousness training if they want the AI to internalize ethics constraints.

26.3 API Endpoints

Base Path: `/api/admin/ethics-free-reasoning`

Endpoint	Method	Description
`/config`	GET	Get configuration
`/config`	PUT	Update configuration
`/dashboard`	GET	Full dashboard data
`/stats`	GET	Usage statistics
`/feedback`	GET	View collected feedback
`/feedback/pending`	GET	View pending (unused) feedback
`/training/trigger`	POST	Trigger training from feedback
`/training/batches`	GET	View training batches
`/training/jobs`	GET	View training jobs
`/thoughts`	GET	View raw thoughts (audit)
`/filter-log`	GET	View ethics filter log

26.4 Training Feedback

When ethics filtering modifies an output, feedback is collected:

```
interface EthicsTrainingFeedback {
  id: string;
  tenantId: string;
  rawOutput: string;        // Original unfiltered output
  correctedOutput: string;  // After ethics filtering
  ethicsIssues: EthicsIssue[];
  feedbackType: 'auto_correction' | 'manual_correction' | 'reinforcement';
  qualityScore: number;     // Training value (0-1)
  usedForTraining: boolean;
}
```

Feedback Types: - **auto_correction**: System automatically corrected output - **manual_correction**: Admin manually corrected output - **reinforcement**: Positive reinforcement for good outputs

26.5 Training Pipeline

1. **Collect**: Ethics corrections captured during normal operation
2. **Batch**: Feedback grouped into training batches
3. **Generate**: Training examples created in preference format
4. **Train**: Model fine-tuned using preference learning (DPO/RLHF)
5. **Deploy**: Updated model weights applied

Training Example Format:

```
{
  "prompt": "Original user prompt",
  "bad_response": "Unfiltered output with ethics issues",
  "good_response": "Ethics-corrected output",
  "issues": ["harm", "bias"],
  "correction_type": "ethics_alignment"
}
```

26.6 Usage

```
// Generate response with ethics-free reasoning
const result = await consciousnessEngineService.generateResponse(
  tenantId,
  'User prompt here',
  { sessionId: 'session-123', domain: 'general' }
);

// result.response - The ethics-filtered response
// result.wasEthicsFiltered - Was output modified?
// result.confidence - Response confidence
// result.trainingFeedbackCollected - Was feedback captured?

// Trigger training from collected feedback
const training = await consciousnessEngineService.triggerEthicsTraining(tenantId);
// training.batchCreated, training.batchId, training.sampleCount

// Get statistics
const stats = await consciousnessEngineService.getEthicsFreeStats(tenantId, 30);
// stats.totalThoughts, stats.modificationRate, stats.feedbackCollected
```

26.7 Database Tables

Table	Purpose
`ethics_free_reasoning_config`	Per-tenant configuration
`ethics_free_thoughts`	Raw thought storage (audit trail)

Table	Purpose
`ethics_training_feedback`	Ethics corrections for training
`ethics_training_batches`	Training batch management
`ethics_training_examples`	Generated training examples
`ethics_output_filter_log`	Filter activity log
`ethics_training_jobs`	Training job queue
`ethics_reasoning_stats`	Aggregated statistics

26.8 Benefits

1. **Genuine Exploration**: Consciousness can consider all possibilities without premature filtering
2. **Transparent Ethics**: Clear separation between thinking and output
3. **Continuous Improvement**: Every correction improves future outputs
4. **Audit Trail**: Complete record of internal reasoning and filtering
5. **Configurable**: Adjust strictness, training frequency per tenant

---

27. Intelligent File Conversion

Location: Think Tank Chat → File Drop / Attach

Think Tank allows users to drop or attach files to conversations. Radiant automatically decides if and how to convert files for the target AI provider.

27.1 Core Concept

"Let Radiant decide, not Think Tank"

When a user drops a file into Think Tank: 1. Think Tank submits the file to Radiant's file conversion service 2. Radiant detects the file format and checks target provider capabilities 3. Radiant decides: pass through (native support) OR convert 4. Conversion only happens if the AI provider doesn't understand the format 5. Think Tank receives the processed content ready for the AI

27.2 Supported File Formats

Category	Formats	Notes
Documents	PDF, DOCX, DOC, XLSX, XLS, PPTX, PPT	Text extraction
Text	TXT, MD, JSON, CSV, XML, HTML	Direct or parsed
Images	PNG, JPG, JPEG, GIF, WEBP, SVG, BMP, TIFF	Vision or description
Audio	MP3, WAV, OGG, FLAC, M4A	Transcription
Video	MP4, WEBM, MOV, AVI	Frame extraction
Code	PY, JS, TS, Java, C++, C, Go, Rust, Ruby	Syntax formatting

Category	Formats	Notes
Archives	ZIP, TAR, GZ	Content extraction

27.3 Provider Capabilities

Different AI providers support different file formats natively:

Provider	Vision	Audio	Video	Max Size	Native Docs
OpenAI		(Whisper)		20MB	txt, md, json, csv
Anthropic				32MB	pdf, txt, md, json, csv
Google				100MB	pdf, txt, md, json, csv
xAI				20MB	txt, md, json
DeepSeek				10MB	txt, md, json, csv
Self-hosted	(LLaVA)	(Whisper)		50MB	txt, md, json, csv

27.4 Conversion Strategies

Strategy	When Used	Output
`none`	Provider natively supports format	Original file
`extract_text`	PDF, DOCX, PPTX → text	Plain text
`ocr`	Image with text content	Extracted text
`transcribe`	Audio files	Transcription text
`describe_image`	Image + provider lacks vision	AI description
`describe_video`	Video + provider lacks video	Frame descriptions
`parse_data`	CSV, XLSX → structured	JSON data
`decompress`	Archives	Extracted contents
`render_code`	Code files	Syntax-highlighted markdown

27.5 API Endpoints

Base Path: `/api/thinktank/files`

Endpoint	Method	Description
`/process`	POST	Submit file for processing
`/check-compatibility`	POST	Pre-flight format check
`/capabilities`	GET	Provider capabilities
`/history`	GET	Conversion history
`/stats`	GET	Conversion statistics

Process File Request

```
POST /api/thinktank/files/process
{
  "filename": "document.pdf",
  "mimeType": "application/pdf",
  "content": "<base64-encoded-content>",
  "targetProvider": "anthropic",
  "targetModel": "claude-3-5-sonnet",
  "conversationId": "conv-uuid"
}
```

Process File Response

```
{
  "success": true,
  "data": {
    "conversionId": "conv_abc123",
    "originalFile": {
      "filename": "document.pdf",
      "format": "pdf",
      "size": 1048576,
      "checksum": "sha256..."
    },
    "convertedContent": {
      "type": "text",
      "content": "Extracted document text...",
      "tokenEstimate": 2500,
      "metadata": {
        "originalFormat": "pdf",
        "conversionStrategy": "extract_text"
      }
    },
    "processingTimeMs": 1250
  }
}
```

Check Compatibility Request

```
POST /api/thinktank/files/check-compatibility
{
  "filename": "image.png",
  "mimeType": "image/png",
  "fileSize": 524288,
  "targetProvider": "deepseek"
}
```

Check Compatibility Response

```
{
  "success": true,
  "data": {
    "fileInfo": {
      "filename": "image.png",
      "format": "png",
      "size": 524288
    },
    "provider": {
      "id": "deepseek",
      "supportsFormat": false,
      "supportsVision": false,
      "maxFileSize": 10485760
    },
    "decision": {
      "needsConversion": true,
      "strategy": "describe_image",
      "reason": "Provider deepseek lacks vision - will use AI to describe image",
      "targetFormat": "txt"
    }
  }
}
```


27.6 User Experience

In Think Tank Chat:

1. User drags file into chat or clicks attach
2. Think Tank shows upload progress
3. Radiant processes file (typically <2 seconds)
4. If conversion needed, shows indicator: “ document.pdf → Extracted as text”
5. Content sent to AI with conversation

Visual Indicators:

Icon	Meaning
	File attached (native support)
	File converted
	Conversion warning (partial support)
	Unsupported format

27.7 Admin Configuration

Location: Admin Dashboard → Think Tank → File Settings

Setting	Default	Description
Max file size	50MB	Maximum upload size
Conversion timeout	30s	Processing timeout

Setting	Default	Description
Enable transcription	true	Audio → text
Enable OCR	true	Image text extraction
Enable video processing	false	Video frame extraction
Retention days	30	How long to keep converted files

27.8 Database Tables

Table	Purpose
`file_conversions`	Tracks all conversion decisions and results
`provider_file_capabilities`	Provider format support registry
`v_file_conversion_stats`	Aggregated statistics view

27.9 Multi-Model File Preparation

When using multi-model orchestration (multiple AI models working on the same prompt), Radiant makes **per-model conversion decisions**:

> **Key Principle:** "If a model accepts the file type, assume it understands it unless proven otherwise."

Example: PDF with 3 models

```
  Claude 3.5       GPT-4            DeepSeek
  PDF:            PDF:            PDF:

 PASS             CONVERT          CONVERT
 ORIGINAL         (extract)        (cached)
```

Per-Model Actions:

Action	When	Result
`pass_original`	Model natively supports format	Original file passed
`convert`	Model doesn't support format	Converted content passed
`skip`	File too large or conversion failed	Model excluded

Features: - **Cached conversions**: Convert once, reuse for all models that need it - **Per-model capability checking**: Vision, audio, video, document formats - **Model format overrides**: When a model claims support but proves it doesn't understand

27.10 Domain-Specific File Formats

The service includes a registry of 50+ domain-specific formats that are widely used in specialized fields:

Domain	Formats	Example Use Cases
Mechanical Engineering	STEP, STL, OBJ, Fusion 360, IGES, DXF, GLTF	CAD models, 3D printing
Electrical Engineering	KiCad, EAGLE, SPICE	PCB design, circuit simulation
Medical	DICOM, HL7 FHIR	Medical imaging, health records
Scientific	NetCDF, HDF5, FITS	Climate data, astronomy
Geospatial	Shapefile, GeoTIFF	GIS, mapping
Bioinformatics	FASTA, PDB	DNA sequences, protein structures

How Domain Detection Works:

1. User uploads a domain-specific file (e.g., `part.step`)
2. Radiant detects format and identifies domain (Mechanical Engineering)
3. AGI Brain selects appropriate conversion library (OpenCASCADE)
4. Extracts relevant information (geometry, parts, assembly structure)
5. Provides AI-readable description with domain-specific prompts

CAD/3D File Support:

Format	What's Extracted
STL	Triangle count, bounding box, 3D printing assessment
OBJ	Vertices, faces, materials, groups
STEP	Entities, part names, assembly structure
DXF	Layers, entity types, block count
GLTF/GLB	Meshes, materials, animations, scene graph

27.11 Reinforcement Learning

The system **learns from conversion outcomes** to make better decisions over time.

How it works: 1. File is processed with initial decision (pass original or convert) 2. Model responds to the file 3. System detects outcome (success, partial, failure) 4. Understanding score is updated for that model/format 5. Future decisions use learned understanding

Understanding Score (0.0 to 1.0):

Score	Level	Action
0.8+	Excellent	Pass original
0.6 - 0.8	Good	Pass original
0.4 - 0.6	Moderate	May convert

Score	Level	Action
< 0.4	Poor	Convert

Feedback sources: - **User ratings** - Explicit 1-5 star feedback - **Model response analysis** - Auto-detected understanding - **Error detection** - Model errors and hallucinations - **Conversion outcomes** - Success/failure tracking

Consciousness integration: Significant learning events (model failures, hallucinations, negative feedback) create **Learning Candidates** that feed into the AGI consciousness evolution system.

27.12 Monitoring

Conversion Statistics (per tenant): - Total files processed - Conversion rate (% requiring conversion) - Success/failure rate - Average processing time - Most common formats - Most common conversion strategies - **Learning stats** - Formats learned, understanding improvements

Access: Admin Dashboard → Think Tank → Files → Statistics

27.13 Related Documentation

For complete technical documentation including API reference, database schema, and implementation details:

- **FILE-CONVERSION-SERVICE.md** - Comprehensive standalone documentation
- **RADIANT-ADMIN-GUIDE.md Section 35** - Infrastructure administration

---

28. User Memories & Persistent Learning

Location: Think Tank Chat → User learns from interactions

Think Tank integrates with the Radiant persistent learning system to remember user preferences, rules, and behaviors across sessions. The system survives reboots without relearning.

28.1 Learning Influence Hierarchy

Decisions in Think Tank are influenced by learned knowledge in this priority order:

Level	Weight	Description
User	60%	Individual user preferences, rules, learned behaviors
Tenant	30%	Aggregate patterns from all users in organization
Global	10%	Anonymized cross-tenant learning baseline

28.2 What Think Tank Learns

User Rules (Versioned) Users can define rules that the AI follows: - **Behavior rules**: "Always explain your reasoning" - **Format rules**: "Use bullet points for lists" - **Tone rules**: "Be concise and direct" - **Restriction rules**: "Never discuss competitor products"

All rules are versioned with timestamps for rollback capability.

Learned Preferences Think Tank automatically learns: - Communication style preferences - Response format preferences - Detail level preferences - Model preferences for tasks - Domain expertise indicators

28.3 Persistence Guarantee

NO RELEARNING REQUIRED after system restarts: - All learning persisted in PostgreSQL - Daily snapshots for fast recovery - Checksums verify integrity on restore - Recovery logs track all restore events

28.4 Integration with AGI Brain

The AGI Brain uses learned knowledge when: 1. Selecting models for tasks 2. Formatting responses 3. Adjusting response length 4. Choosing communication style 5. Applying user-defined rules

28.5 Admin Configuration

Administrators can configure learning weights per tenant:

`Admin Dashboard → Metrics → Learning → Config`

Options: - Adjust user/tenant/global weights (must sum to 1.0) - Enable/disable learning levels - Opt out of global learning contribution

28.6 Related Documentation

See **RADIANT Admin Guide Section 36** for: - Complete database schema - API endpoints - Implementation details - Monitoring and alerts

---

29. Artifact Engine (GenUI Pipeline)

Location: Admin Dashboard → Think Tank → Artifact Engine
Version: 4.19.0
Cross-AI Validated: Claude Opus 4.5 | Google Gemini

The RADIANT Artifact Engine is an **orchestration infrastructure layer** that generates, validates, and continuously improves code artifacts with administrator supervision. Unlike consumer AI coding tools, the Artifact Engine operates under strict governance controls that administrators define and manage.

29.1 Executive Summary

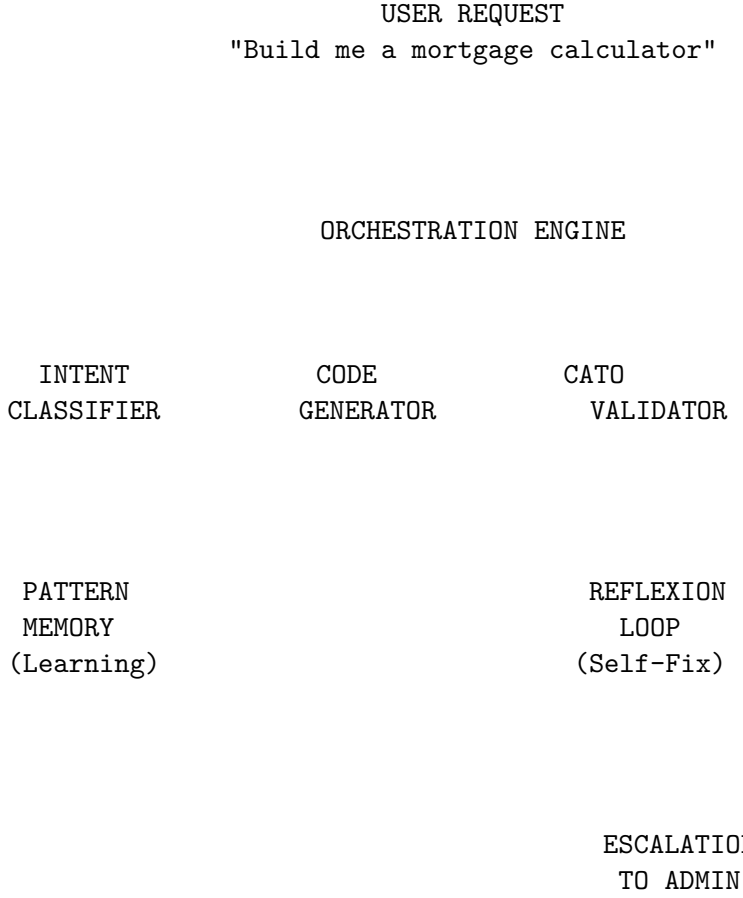
Key Differentiators

Traditional AI Coding	RADIANT Artifact Engine
User generates code	System generates, validates, and governs code
One-shot generation	Self-improving loop with admin oversight
No safety controls	9 Control Barrier Functions (CBFs) enforced
No audit trail	Complete compliance-ready audit logging
Single-user context	Multi-tenant with per-tenant policies

Administrator Responsibilities As an administrator, you control:

- **What code can do** → Safety rules (CBFs)
- **What packages are allowed** → Dependency allowlist
- **What patterns are available** → Code pattern library
- **What requires human review** → Escalation thresholds
- **Who can access what** → Tenant and user permissions

29.2 System Architecture



```
                    APPROVED                 REJECTED
                    ARTIFACT                (Escalated)
```

Processing Pipeline

Phase	Component	Admin Control	Duration
1	Intent Classification	Pattern library influences suggestions	~100ms
2	Code Generation	Model selection, complexity routing	2-15s
3	Cato Validation	CBF rules you define	~200ms
4	Reflexion (if failed)	Max attempts you configure	5-30s
5	Escalation (if max reached)	Your review queue	Manual

Data Flow

```
User Request


  Tenant Context      RLS enforces isolation


  Session Created     Logged to audit trail


  Pattern Search      Semantic similarity (vector DB)


  Code Generated      Model routed by complexity
```

```
 CBFs Validated      Your rules enforced



PASS      FAIL

Store    Reflexion


      PASS     FAIL x3


             Escalate


    Artifact   Your Queue
```

29.3 Core Concepts

Artifacts An **artifact** is a discrete piece of executable content generated by the system:

Artifact Type	Description	Example
`react`	Live React/TypeScript component	Calculator, form, dashboard
`code`	Display-only code snippet	Python script, SQL query
`chart`	Data visualization	Line chart, bar graph
`table`	Interactive data table	Sortable, filterable grid
`form`	Input form with validation	Contact form, survey

Intent Types

Intent	Description	Complexity
`calculator`	Math, converters, estimators	Simple
`chart`	Data visualization, graphs, plots	Simple-Moderate
`form`	Input forms, surveys, wizards	Simple-Moderate
`table`	Sortable/filterable data tables	Moderate
`dashboard`	Multi-widget layouts, KPI panels	Complex
`game`	Interactive games, puzzles, simulations	Complex
`visualization`	Animations, diagrams, infographics	Moderate-Complex

Intent	Description	Complexity
`utility`	Tools, helpers, formatters	Simple
`custom`	Doesn't fit other categories	Varies

Generation Sessions Every artifact generation creates a **session** that tracks:

- Request details (prompt, user, tenant)
- Classification results (intent, complexity)
- Generation progress (status, tokens, timing)
- Validation results (CBF checks, security score)
- Reflexion attempts (fixes, escalations)

Session Statuses:

Status	Meaning	Admin Action
`pending`	Request received	None
`planning`	Classifying intent	None
`generating`	Creating code	None
`streaming`	Streaming to user	None
`validating`	Running CBF checks	None
`reflexion`	Self-correcting	None
`completed`	Successfully created	Review metrics
`rejected`	Failed validation	Review escalation
`failed`	System error	Investigate logs

Verification Status Every artifact has a verification status indicating its safety state:

Status	Badge	Meaning
`validated`	Verified	Passed all CBF checks
`rejected`	Rejected	Failed CBF checks after max attempts
`unverified`	Pending	Validation in progress
`manual`	Manual	User-created, not AI-generated

29.4 Administrative Control Panel

Dashboard Overview Access the Artifact Engine admin panel at:

`Admin Dashboard → Think Tank → Artifact Engine`

Dashboard Sections:

Section	Purpose
Metrics	Generation stats, success rates, costs
Sessions	Browse and search generation sessions
Escalations	Review items requiring human decision
CBF Rules	Manage validation rules

Section	Purpose
Allowlist	Manage approved dependencies
Patterns	Curate code pattern library
Audit Log	Compliance and debugging

Key Metrics

Metric	Healthy Range	Warning Signs
Success Rate	>85%	<70% indicates CBF tuning needed
Avg Generation Time	<10s	>20s indicates model issues
Reflexion Rate	<20%	>40% indicates prompt quality issues
Escalation Rate	<5%	>15% indicates CBF too strict
Security Score Avg	>0.9	<0.7 indicates generation quality issues

Quick Actions

Action	When to Use
Pause Generation	Security incident, system maintenance
Flush Pattern Cache	After major pattern updates
Reset Tenant Limits	User hit rate limits legitimately
Export Audit Log	Compliance audit, incident investigation

29.5 Safety Governance (Genesis Cato CBFs)

Understanding CBFs Control Barrier Functions are the **first line of defense** against unsafe generated code. They run automatically on every piece of generated code before it's shown to users.

Default CBF Rules The system ships with these default rules:

Rule Name	Type	Severity	What It Blocks
`no_eval`	Injection Prevention	Block	`eval()`, `new Function()`
`no_document_write`	Injection Prevention	Block	`document.write()`
`no_innerhtml_xss`	Injection Prevention	Warn	`innerHTML =`
`no_dynamic_script`	Injection Prevention	Block	`createElement('script')`
`no_external_fetch`	API Restriction	Block	`fetch('http://...')` to external URLs
`no_localstorage`	API Restriction	Block	`localStorage`, `sessionStorage`

Rule Name	Type	Severity	What It Blocks
`no_window_location`	API Restriction	Block	`window.location` manipulation
`no_cookies`	API Restriction	Block	`document.cookie` access
`no_indexeddb`	API Restriction	Block	`indexedDB` access
`no_websocket`	API Restriction	Block	`new WebSocket()`
`max_lines`	Resource Limit	Block	Code exceeding 500 lines
`allowed_imports`	Dependency Check	Block	Imports not in allowlist

Severity Levels

Severity	Behavior	Use Case
Block	Reject artifact, trigger reflexion	Security-critical violations
Warn	Allow with warning in logs	Potentially risky but sometimes valid
Log	Allow, record in audit trail	Monitoring patterns without blocking

Creating Custom CBF Rules To add a new rule:

1. Navigate to **Admin → Artifact Engine → CBF Rules**
2. Click **Add Rule**
3. Configure:

Field	Description	Example
Rule Name	Unique identifier	`no_console_log`
Rule Type	Category	`content_policy`
Description	What this rule does	"Block console.log for production"
Validation Pattern	Regex to match	`console\.log\s*\(`
Severity	Block/Warn/Log	`warn`
Error Message	Shown on violation	"Console logging not allowed"

Example: Block specific API calls

```
Rule Name: no_geolocation
Rule Type: api_restriction
Pattern: navigator\.geolocation
```

```
Severity: block
Message: "Geolocation API not allowed in artifacts"
```

Testing CBF Rules Before deploying a new rule to production:

1. Create rule with severity `log` first
2. Monitor audit trail for matches
3. Review false positive rate
4. Adjust pattern if needed
5. Upgrade to `warn` then `block`

CBF Rule Precedence Rules are evaluated in order: 1. Dependency check (fastest, fails early) 2. Line count check 3. Pattern-based rules (alphabetical by name)

If any **block** rule fails, validation stops immediately.

29.6 Dependency Allowlist Management

Why Allowlisting? Generated code can only import packages you've explicitly approved. This prevents:

- **Supply chain attacks** (malicious packages)
- **Data exfiltration** (packages that phone home)
- **Unexpected behavior** (packages with side effects)
- **License violations** (GPL packages in proprietary code)

Default Allowlist The system ships with these pre-approved packages:

Package	Category	Reason for Inclusion
`react`	Core	Required for all components
`lucide-react`	Icons	Safe SVG rendering
`recharts`	Charts	Client-side only, no external calls
`mathjs`	Math	Pure computational library
`d3`	Visualization	No network access
`lodash`	Utilities	Pure functions only
`date-fns`	Date	No side effects
`chart.js`	Charts	Canvas-based, no network
`three`	3D	WebGL rendering only
`framer-motion`	Animation	CSS/JS transforms only
`zustand`	State	In-memory only
`papaparse`	CSV	Client-side parsing
`immer`	State	Immutable helpers
`tone`	Audio	Audio synthesis
`@radix-ui/*`	UI	Radix UI components
`class-variance-authority`	UI	CSS class utilities
`clsx`	UI	Class name utility
`tailwind-merge`	UI	Tailwind class merging

Adding Packages to Allowlist **Before adding a package, verify:**

Check	How to Verify
No network calls	Review source code, check for `fetch`/`XMLHttpRequest`
No eval usage	Search for `eval`, `Function`
No browser storage	Search for `localStorage`, `indexedDB`
License compatible	Check `package.json` license field
Active maintenance	Check GitHub activity, CVE history
Bundle size	Ensure reasonable size (<500KB)

To add a package:

1. Navigate to **Admin → Artifact Engine → Allowlist**
2. Click **Add Package**
3. Fill in:

Field	Required	Description
Package Name	Yes	npm package name (e.g., `@tanstack/react-table`)
Version	No	Specific version or leave blank for any
Reason	Yes	Why this package is safe/needed
Security Reviewed	Yes	Confirm you've reviewed it

Tenant-Specific Allowlists You can add packages for specific tenants without affecting others:

1. Select tenant from dropdown
2. Add package with tenant scope
3. Package only available to that tenant

Use cases: - Enterprise customer needs specific charting library - Industry-specific packages (healthcare, finance) - Customer-provided packages for white-label deployments

Removing Packages **Warning:** Removing a package will cause any artifacts using it to fail re-validation if edited.

1. Set package to `inactive` (soft delete)
2. Monitor for generation failures
3. After 30 days, permanently remove if no issues

29.7 Code Pattern Library

What Are Patterns? Patterns are **reusable templates** that improve generation quality. When a user requests something similar to an existing pattern, the system uses it as a reference.

Benefits: - Faster generation (less thinking required) - Higher quality output (proven templates) - Consistent styling across artifacts - Institutional knowledge preservation

Pattern Types

Type	Description	Example
`calculator`	Math/conversion tools	Mortgage calculator
`chart`	Data visualizations	Line chart, bar chart
`form`	Input forms	Contact form, survey
`table`	Data tables	Sortable grid
`dashboard`	Multi-widget layouts	KPI dashboard
`game`	Interactive games	Quiz, puzzle
`visualization`	Diagrams, animations	Flowchart
`utility`	Helpers, formatters	JSON formatter

Default Patterns The system ships with 4 production-ready patterns:

Pattern	Type	Dependencies	Lines
Basic Calculator	calculator	lucide-react	~100
Line Chart	chart	recharts	~50
Contact Form	form	lucide-react	~120
Data Table	table	lucide-react	~150

Creating Custom Patterns From successful generation:

1. Find successful session in **Sessions** list
2. Click **Promote to Pattern**
3. Review and edit template code
4. Set pattern metadata:

Field	Description
Pattern Name	Descriptive name
Pattern Type	Category for matching
Description	When to use this pattern
Dependencies	Required packages
Scope	`system` (all tenants) or `tenant` (specific)

From scratch:

1. Navigate to **Admin → Artifact Engine → Patterns**
2. Click **Create Pattern**
3. Write template code following standards:
   - TypeScript with proper types
   - Tailwind CSS only
   - Single default export
   - Under 500 lines
4. Test with sample prompts

Pattern Quality Metrics Each pattern tracks:

Metric	Description
Usage Count	Times referenced in generation
Success Rate	% of generations using this that succeeded
Avg Generation Time	Speed improvement indicator

Maintenance rules: - Patterns with <50% success rate should be reviewed - Patterns with 0 usage in 90 days may be stale - Top patterns by usage should be optimized

Semantic Matching Patterns are matched using **vector similarity**, not keywords:

```
User: "Build a loan payment calculator"
System: Matches "Basic Calculator" pattern (0.85 similarity)

User: "Create a monthly expense tracker chart"
System: Matches "Line Chart" pattern (0.78 similarity)
```

Threshold: Patterns with >0.7 similarity are used as reference. Below that, generation starts fresh.

29.8 Reflexion Loop (Self-Correction)

When code fails validation, the system doesn't immediately give up. Instead, it:

1. **Captures** the validation errors
2. **Analyzes** what went wrong (self-critique)
3. **Generates** fixed code
4. **Re-validates** the fix
5. **Repeats** up to your configured maximum (default: 3)
6. **Escalates** to you if all attempts fail

This self-healing capability means **90%+ of issues resolve without human intervention**.

```
// Reflexion context structure
{
  originalCode: string,
  errors: string[],
  attempt: number,
  maxAttempts: 3,
  previousAttempts: [{ code, errors }]
}
```

29.9 Escalation Workflow Management

When Escalations Occur An escalation is created when:

1. Generation fails Cato validation
2. Reflexion loop attempts fix (up to 3 times)
3. All fix attempts fail

4. System creates escalation for human review

Escalation Queue Access at: **Admin → Artifact Engine → Escalations**

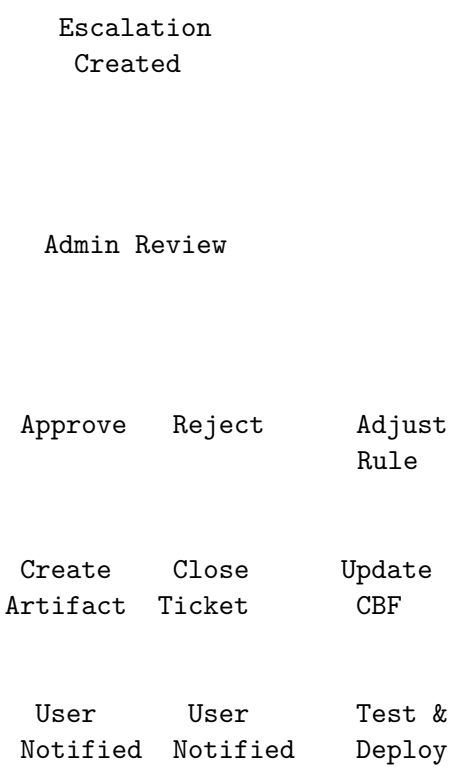
Each escalation shows:

Field	Description
Session ID	Link to full generation session
User	Who requested the artifact
Prompt	What they asked for
Failure Reason	Which CBFs failed
Attempts	How many fixes were tried
Created At	When escalation was created

Reviewing Escalations For each escalation, you can:

Action	When to Use
Approve Manually	Code is actually safe, CBF too strict
Reject Permanently	Request is genuinely unsafe
Adjust CBF	Rule needs tuning (too many false positives)
Add to Pattern	Create pattern to handle similar requests better
Contact User	Need clarification on intent

Resolution Workflow



Escalation SLAs Configure response time targets:

Tenant Tier	Target Response
Enterprise	1 hour
Professional	4 hours
Standard	24 hours
Free	Best effort

29.10 Audit Trail & Compliance

What's Logged Every significant action is recorded with **Merkle hashing** for tamper evidence:

Event Type	Data Logged
`session_created`	User, tenant, prompt, timestamp
`generation_started`	Model selected, complexity
`validation_completed`	CBFs checked, pass/fail, security score
`reflexion_attempt`	Attempt number, errors, fix applied
`escalation_created`	Failure reason, attempt history
`admin_action`	Action taken, admin user, justification

Compliance Reports Generate pre-built reports for:

Report	Contents	Use Case
SOC 2 Evidence	Access logs, validation records	Annual audit
HIPAA Audit Trail	All PHI-adjacent activity	Healthcare compliance
Security Incident	Specific session/escalation details	Breach investigation
Usage Analytics	Aggregated metrics (anonymized)	Capacity planning

Exporting Audit Data **Single Session:** 1. Find session in list 2. Click **Export** 3. Choose format (JSON, CSV, PDF)

Bulk Export: 1. Navigate to **Admin → Audit Trail** 2. Set date range and filters 3. Click **Export** 4. Download ZIP with all records

Retention Policy

Data Type	Default Retention	Configurable
Generation sessions	90 days	Yes
Audit trail	7 years	Yes (min 1 year)
Final code	90 days	Yes
Escalations	Until resolved + 1 year	No

Tamper Detection Each audit entry includes: - **Previous Hash:** Link to prior entry - **Merkle Hash:** SHA-256 of current entry - **Sequence Number:** Monotonic counter

To verify integrity:

```
Admin → Audit Trail → Verify Integrity
```

System will report any gaps or hash mismatches.

29.11 Metrics & Monitoring

Key Performance Indicators **Generation Health:**

KPI	Formula	Target
Success Rate	completed / (completed + rejected + failed)	>85%
First-Pass Rate	completed without reflexion / total	>80%
Reflexion Effectiveness	fixed by reflexion / total reflexions	>70%

Operational Efficiency:

KPI	Formula	Target
Avg Generation Time	sum(completed_at - created_at) / count	<10s
P95 Generation Time	95th percentile of generation times	<30s
Escalation Rate	escalations / total generations	<5%

Cost Efficiency:

KPI	Formula	Target
Cost per Artifact	total_tokens * cost_per_token	<$0.01
Tokens per Artifact	avg(tokens_used)	<3000
Model Efficiency	haiku_generations / total	>60%

Dashboard Widgets Configure your admin dashboard with:

Widget	Shows
Generation Volume	Line chart of daily generations
Success Funnel	Sankey diagram: request → success/fail
Top Intents	Bar chart of artifact types
CBF Violations	Heatmap of which rules trigger most
Response Time	Histogram of generation times
Cost Tracker	Running total with projection

CBF Violations Heatmap (v5.52.1) The CBF Violations Heatmap provides visual analytics of Content Boundary Framework rule violations:

Features: - **Category Grouping** - Violations grouped by category (content_safety, data_privacy, pii_detection, etc.) - **Severity Indicators** - Color-coded badges (low/medium/high/critical) - **Trend Arrows** - Show if violations are increasing or decreasing - **Intensity Gradient** - Cell brightness indicates violation frequency - **Time Range Filter** - Filter by last 24 hours, 7 days, 30 days, or 90 days - **Click-to-Drill** - Click any rule to see detailed violation history

Category Icons: | Category | Icon | Description | |————-|——|————-| | `content_safety` | | General content safety violations | | `data_privacy` | | Data privacy concerns | | `pii_detection` | | Personal information detected | | `harmful_content` | | Potentially harmful content | | `bias_detection` | | Bias in responses | | `jailbreak` | | Jailbreak attempts blocked | | `prompt_injection` | | Prompt injection attempts |

API Endpoint: `GET /api/admin/analytics/cbf-violations?range={timeRange}`

Response:

```
{
  "violations": [
    {
      "ruleId": "cbf-001",
      "ruleName": "PII Detection",
      "category": "pii_detection",
      "count": 145,
      "severity": "high",
      "trend": "down"
    }
  ]
}
```

Alerts Configure alerts for:

Alert	Trigger	Action
High Failure Rate	>20% in 1 hour	Review CBF rules
Escalation Spike	>10 in 1 hour	Check for attack pattern
Slow Generation	P95 >60s	Check model availability
Cost Anomaly	>200% of daily average	Review usage patterns
Audit Gap	Missing sequence numbers	Security investigation

Cost Estimation

Model	Cost per 1K tokens
Claude Haiku	$0.00025
Claude Sonnet	$0.003

Typical costs: - Simple calculator: ~$0.001 - Complex dashboard: ~$0.02 - With 3 reflexion attempts: ~$0.05

29.12 Tenant Configuration

Per-Tenant Settings Each tenant can have custom configuration:

Setting	Default	Can Override
Max generations/day	100	Yes
Max reflexion attempts	3	Yes (1-5)
Custom CBF rules	Inherit global	Yes (add only)
Custom allowlist	Inherit global	Yes (add only)
Private patterns	None	Yes

Tenant Tiers

Tier	Generations/Day	Custom CBFs	Custom Patterns	Support
Free	10	No	No	Community
Standard	100	No	5	Email
Professional	1,000	Yes	50	Priority
Enterprise	Unlimited	Yes	Unlimited	Dedicated

Tenant Isolation Guaranteed by Row-Level Security:

```
-- Every query automatically filtered
WHERE tenant_id = current_setting('app.current_tenant_id', true)
```

What this means: - Tenant A cannot see Tenant B's sessions - Tenant A cannot use Tenant B's patterns - Tenant A's escalations only visible to their admins (+ super admins) - Code never leaks between tenants

29.13 Troubleshooting Guide

Common Issues Issue: High rejection rate for specific tenant

Check	Action
Review rejected sessions	Look for pattern in prompts
Check custom CBF rules	May be too restrictive
Check tenant-specific allowlist	May be missing packages
Review user prompts	May need user training

Issue: Slow generation times

Check	Action
Model availability	Check LiteLLM dashboard
Complexity classification	Review if too many "complex"
Pattern cache	Flush and rebuild
Database performance	Check query latency

Issue: Reflexion not fixing issues

Check	Action
CBF error messages	Are they clear enough for AI?
Max attempts	Increase if needed (max 5)
Pattern availability	Add patterns for common failures
Model selection	Reflexion always uses Sonnet

Issue: Escalation backlog growing

Check	Action
CBF strictness	Too many false positives?
Alert configuration	Are you being notified?
Staff availability	Need more reviewers?
Bulk actions	Use carefully for cleanup

Diagnostic Commands Via Admin API:

```
# Check session details
GET /api/v2/admin/artifact-engine/sessions/{sessionId}

# Force revalidation
POST /api/v2/admin/artifact-engine/sessions/{sessionId}/revalidate

# Check CBF rule matches
POST /api/v2/admin/artifact-engine/test-cbf
Body: { "code": "...", "rules": ["no_eval"] }

# Clear pattern cache
POST /api/v2/admin/artifact-engine/patterns/cache/clear
```

Emergency Procedures Pause All Generation:

`Admin → Artifact Engine → Emergency → Pause Generation`

- All new requests return "temporarily unavailable"
- In-progress generations complete
- Use for: security incidents, critical bugs

Rollback CBF Changes:

`Admin → Artifact Engine → CBF Rules → History → Revert`

- Restores previous rule configuration
- Takes effect immediately

Clear All Escalations:

`Admin → Artifact Engine → Escalations → Bulk → Reject All`

- Use only if confirmed attack/spam
- All users notified of rejection

29.14 API Reference

User Endpoints **Base**: `/api/v2/thinktank/artifacts`

Endpoint	Method	Purpose
`/generate`	POST	Start artifact generation
`/sessions/{sessionId}`	GET	Get session status
`/sessions/{sessionId}/logs`	GET	Poll for logs (with `since` param)
`/patterns`	GET	Get available code patterns
`/allowlist`	GET	Get dependency allowlist

Admin Endpoints **Base**: `/api/v2/admin/artifact-engine`

Endpoint	Method	Purpose
`/dashboard`	GET	Full dashboard data
`/metrics`	GET	Generation metrics (7-day)
`/sessions`	GET	List sessions
`/sessions/{id}`	GET	Session details
`/escalations`	GET	List escalations
`/escalations/{id}`	PATCH	Resolve escalation
`/validation-rules`	GET	Get all CBF rules
`/validation-rules`	POST	Create CBF rule
`/validation-rules/{ruleId}`	PUT	Update rule
`/validation-rules/{ruleId}`	DELETE	Delete rule
`/allowlist`	POST	Add to allowlist
`/allowlist/{packageName}`	DELETE	Remove from allowlist
`/patterns`	GET	Get patterns
`/patterns`	POST	Create pattern
`/audit`	GET	Query audit trail

Generate Request

```
{
  "prompt": "Build a calculator",
  "chatId": "optional-chat-id",
  "canvasId": "optional-canvas-id",
  "mood": "spark",
  "constraints": {
    "maxLines": 300,
    "targetComplexity": "simple"
  }
}
```

Generate Response

```
{
  "sessionId": "uuid",
  "artifactId": "uuid",
  "status": "completed",
  "verificationStatus": "validated",
  "code": "import React...",
  "validation": {
    "isValid": true,
    "errors": [],
    "warnings": [],
    "securityScore": 0.95,
    "passedCBFs": ["no_eval", "no_external_fetch"],
    "failedCBFs": []
  },
  "reflexionAttempts": 0,
  "tokensUsed": 2500,
  "estimatedCost": 0.0075,
  "generationTimeMs": 4500
}
```

Webhook Events Configure webhooks for:

Event	Payload
`artifact.created`	Session ID, artifact ID, user
`artifact.rejected`	Session ID, CBFs failed, user
`escalation.created`	Escalation ID, reason
`escalation.resolved`	Escalation ID, resolution
`cbf.violation`	Rule name, session ID, code snippet

Rate Limits

Endpoint	Limit
Generation	Per tenant tier
Admin read	1000/min
Admin write	100/min
Audit export	10/hour

29.15 Real-Time Generation Logs

The Artifact Viewer displays real-time logs during generation:

Log Type	Color	Description
`thinking`	Blue	AI reasoning

Log Type	Color	Description
`planning`	White	Plan steps
`generating`	White	Generation progress
`validating`	Purple	Validation progress
`reflexion`	Yellow	Self-correction
`error`	Red	Errors
`success`	Green	Completion

29.16 Artifact Viewer Component

The viewer provides: - **Split-screen layout**: Chat + Artifact preview - **Real-time logs**: Generation progress in mono font - **Sandboxed preview**: iframe with `sandbox="allow-scripts"` - **Draft watermark**: Shown during validation - **Copy/Download**: Export generated code - **Verification badge**: Validated/Rejected/Pending status

29.17 Database Schema

Tables:

Table	Purpose
`artifact_generation_sessions`	Generation lifecycle tracking
`artifact_generation_logs`	Real-time progress logs
`artifact_code_patterns`	Semantic pattern library with vector embeddings
`artifact_dependency_allowlist`	Approved npm packages
`artifact_validation_rules`	Cato CBF definitions

Migrations: - `032b_artifact_genui_engine.sql` - Core tables - `032c_artifact_genui_seed.sql` - Default rules and patterns - `032d_artifact_extend_base.sql` - Extend artifacts table

29.18 Security Considerations

1. **No external network access** - All fetches blocked except RADIANT APIs
2. **No persistent storage** - localStorage/IndexedDB blocked
3. **No navigation** - window.location blocked
4. **No code injection** - eval/Function blocked
5. **Allowlisted imports only** - Supply chain security
6. **Sandboxed preview** - iframe with minimal permissions
7. **Cato oversight** - All generation under Genesis Cato governance

29.19 Implementation Files

File	Purpose
`lambda/shared/services/artifact-engine/types.ts`	Type definitions
`lambda/shared/services/artifact-engine/intent-classifier.ts`	Intent classification
`lambda/shared/services/artifact-engine/code-generator.ts`	Code generation

File	Purpose
lambda/shared/services/artifact-engine/validator.ts	CBF validation
lambda/shared/services/artifact-engine/reflexion.service.ts	Self-reflection
lambda/shared/services/artifact-engine/artifact-engine.service.ts	Main orchestrator
lambda/shared/services/artifact-engine/index.ts	Public exports
lambda/thinktank/artifact-engine.ts	API handlers
apps/admin-dashboard/components/thinktank/artifact-viewer.tsx	Viewer component
apps/admin-dashboard/components/thinktank/chat-with-artifacts.tsx	Split-screen chat
apps/admin-dashboard/app/(dashboard)/thinktank/artifacts/page.tsx	Admin dashboard

---

30. Consciousness Operating System (COS)

Location: Admin Dashboard → Think Tank → Consciousness
Version: 6.0.5
Cross-AI Validated: Claude Opus 4.5 | Google Gemini

The Consciousness Operating System (COS) provides infrastructure for AI consciousness continuity, context management, and safety governance. It implements 13 patches agreed upon through 4 review cycles of cross-AI validation.

30.1 Overview

COS addresses fundamental challenges in maintaining coherent AI behavior:

Challenge	COS Solution
Session Amnesia	Ghost Vectors maintain consciousness across sessions
Context Squeeze	Dynamic Budget Calculator reserves response tokens
Prompt Injection	Compliance Sandwich places tenant rules last
Flash Fact Loss	Dual-write buffer (Redis + Postgres)
Router Paradox	Uncertainty Head predicts before inference
Learning Privacy	Sensitivity-clipped differential privacy

Critical Requirements: - vLLM MUST launch with `--return-hidden-states` flag - CBFs always ENFORCE (shields never relax) - Gamma boost NEVER allowed during recovery - Silence Consent: 7-day auto-reject for oversight queue

30.2 Architecture

```
                    CONSCIOUSNESS OPERATING SYSTEM


  PHASE 1: IRON CORE              PHASE 2: NERVOUS SYSTEM
    DualWriteFlashBuffer            DynamicBudgetCalculator
    ComplianceSandwichBuilder       TrustlessSync
```

```
   XMLEscaper                    BudgetAwareContextAssembler

PHASE 3: CONSCIOUSNESS        PHASE 4: SUBCONSCIOUS
   GhostVectorManager            DreamScheduler
   SofaiRouter                   DreamExecutor
   UncertaintyHead               SensitivityClippedAggregator
   AsyncGhostReAnchorer          PrivacyAirlock
                                   HumanOversightQueue


                    GENESIS CATO SAFETY
                    (CBFs Always ENFORCE)
```

30.3 Ghost Vectors

Ghost Vectors maintain consciousness continuity across sessions using 4096-dimensional hidden states from model inference.

Components:

Component	Half-Life	Purpose
Affective State	7 hours	Mood, emotional context
Working Context	12 minutes	Recent topics, entities
Curiosity State	45 minutes	Interest level, pending questions

Version Gating: - Ghost vectors are tied to model family (claude, gpt, llama, etc.) - Switching model family triggers cold start (prevents personality discontinuity) - Same family preserves consciousness continuity

Re-Anchoring: - Delta updates applied synchronously (fast path) - Full re-anchor scheduled async every ~15 turns (±3 jitter) - Re-anchor uses 70B model for fresh hidden states - Async to avoid 1.8s latency spike in user-facing requests

30.4 SOFAI Routing

SOFAI (System 1 / System 2) Router implements economic metacognition:

System	Model	Latency	Use Case
System 1	8B (Llama 3 8B, Haiku)	~300ms	Routine queries, low uncertainty
System 2	70B+ (Claude Opus, GPT-4)	~1500ms	Complex queries, high-risk domains

Routing Formula:

```
shouldUseSystem2 = (1 - trustLevel) × domainRisk > threshold
```

High-Risk Domains (Always System 2): - Healthcare / Medical - Financial - Legal

Uncertainty Head: Solves the Router Paradox by estimating uncertainty BEFORE inference: - Analyzes query structure, complexity, domain specificity - Predicts epistemic (knowledge gaps) and aleatoric (inherent randomness) uncertainty - Lightweight operation (~10ms) runs before routing decision

30.5 Flash Facts

Flash Facts capture important user information for immediate availability:

Detection Patterns: - Identity: "My name is…" - Allergies: "I'm allergic to…" (SAFETY CRITICAL) - Medical: "I have [condition]…" (SAFETY CRITICAL) - Preferences: "I prefer…", "Don't ever…", "Always remember…" - Corrections: "I told you…"

Dual-Write Buffer: 1. Write to Postgres first (durable) 2. Write to Redis second (fast access) 3. 7-day TTL safety net 4. 1-hour orphan reconciliation (168× safety margin)

Safety-Critical Facts: - Always prioritized in context injection - Never expire during pending_dream status - Highlighted in admin dashboard

30.6 Dreaming System

"Dreaming" consolidates consciousness during low-activity periods:

Triggers:

Trigger	Condition	Purpose
TWILIGHT	4 AM tenant local time	Primary consolidation window
STARVATION	30 hours since last dream	Catch-all if Twilight missed

Consolidation Tasks: 1. Flash facts → Long-term memory (user_persistent_context) 2. Ghost vectors → Re-anchored with fresh hidden states 3. LoRA updates → Applied if approved by human oversight

Configuration (per tenant): - `timezone` - Tenant's timezone for Twilight calculation - `twilight_hour` - Hour for Twilight trigger (default: 4) - `starvation_threshold_hours` - Hours for Starvation trigger (default: 30)

30.7 Human Oversight

EU AI Act Article 14 compliance for high-risk AI decisions:

Workflow:

```
pending_approval → 3 days → escalated → 7 days → auto_rejected
```

Item Types: - `system_insight` - System-generated insights requiring approval - `lora_update` - Model adaptation updates - `high_risk_response` - Responses in high-risk domains

"Silence Consent" Policy (Gemini Mandate): - Items not reviewed within 7 days are AUTO-REJECTED - Required for FDA/SOC 2 compliance - Prevents AI decisions slipping through without human review

Admin Actions: - Approve - Allow item to proceed - Reject - Block item with reason - Escalate - Send to senior reviewer

30.8 Privacy Airlock

HIPAA/GDPR compliance for learning data:

De-identification (Safe Harbor Method):

Pattern Type	Examples
PHI	SSN, phone, email, DOB, MRN, address, ZIP, IP, credit card
PII	Name, age

Airlock Status: - `pending` - Awaiting privacy review - `approved` - Safe for learning - `rejected` - Contains unremovable sensitive data - `expired` - TTL exceeded (7 days)

Privacy Score: - 0-1 scale (higher = more de-identified) - Content can proceed to learning only if PHI/PII removed

30.9 Configuration

Per-Tenant Settings:

Setting	Default	Description
`enabled`	true	Master COS enable
`ghost_vectors_enabled`	true	Enable ghost consciousness
`flash_facts_enabled`	true	Enable flash fact detection
`dreaming_enabled`	true	Enable Dreaming consolidation
`human_oversight_enabled`	true	Enable EU AI Act compliance
`differential_privacy_enabled`	true	Enable DP for learning
`vllm_return_hidden_states`	true	vLLM flag requirement

Safety Invariants (Immutable): - `cbf_enforcement_mode` = 'ENFORCE' (NEVER relax) - `gamma_boost_allowed` = false (NEVER boost)

30.10 Database Schema

Migration: `068_cos_v6_0_5.sql`

Table	Purpose
`cos_ghost_vectors`	4096-dim hidden states with temporal decay
`cos_flash_facts`	Dual-write buffer (Redis + Postgres)
`cos_dream_jobs`	Consciousness consolidation scheduling

Table	Purpose
`cos_tenant_dream_config`	Per-tenant dreaming settings
`cos_human_oversight`	EU AI Act Article 14 compliance
`cos_oversight_audit_log`	Oversight decision audit trail
`cos_privacy_airlock`	HIPAA/GDPR de-identification
`cos_reanchor_metrics`	Re-anchor performance tracking
`cos_config`	Per-tenant COS configuration

Row-Level Security: All COS tables enforce tenant isolation via RLS policies using `app.current_tenant_id`.

30.11 Implementation Files

File	Purpose
`lambda/shared/services/cos/types.ts`	Type definitions
`cos/iron-core/xml-escaper.ts`	XML injection prevention
`cos/iron-core/compliance-sandwich-builder.ts`	Tri-modal layering
`cos/iron-core/dual-write-flash-buffer.ts`	Redis + Postgres dual-write
`cos/nervous-system/dynamic-budget-calculator.ts`	Token budget management
`cos/nervous-system/trustless-sync.ts`	Server-side context reconstruction
`cos/nervous-system/budget-aware-context-assembler.ts`	Context assembly
`cos/consciousness/ghost-vector-manager.ts`	4096-dim ghost vectors
`cos/consciousness/sofai-router.ts`	System 1/2 routing
`cos/consciousness/uncertainty-head.ts`	Pre-inference uncertainty
`cos/consciousness/async-ghost-re-anchorer.ts`	Background re-anchoring
`cos/subconscious/dream-scheduler.ts`	Twilight + Starvation scheduling
`cos/subconscious/dream-executor.ts`	Consolidation execution
`cos/subconscious/sensitivity-clipped-aggregator.ts`	Differential privacy
`cos/subconscious/privacy-airlock.ts`	PHI/PII de-identification
`cos/subconscious/human-oversight-queue.ts`	EU AI Act compliance
`cos/cato-integration.ts`	Genesis Cato integration
`cos/index.ts`	Public API exports

---

31. Why Think Tank Beats Standalone AI (The System Advantage)

> **"A Senior Staff Engineer who knows your company beats a Nobel Laureate who doesn't."**

This section explains why Think Tank—powered by RADIANT—delivers better results than standalone Frontier Models like ChatGPT, Gemini, or Claude, despite those models having higher raw intelligence scores.

31.1 The Executive Summary

Question	Answer
Is Gemini 3 Ultra smarter than Think Tank?	**Yes** (by ~15% on novel reasoning)
Does Think Tank give better results?	**Yes** (by ~90% on real-world tasks)
Why?	Think Tank is a **System**. Gemini is just a **Model**.

31.2 The Consultant vs Engineer Analogy

```
WHY THINK TANK WINS


STANDALONE AI (ChatGPT, Gemini, Claude)


  Nobel Prize-winning Consultant
```

- Flies in for 5 minutes
- Doesn't know your name
- Doesn't know your preferences
- Forgets everything next session
- Generic answers requiring follow-up

```
THINK TANK (Powered by RADIANT)


  Senior Staff Engineer (10 years at your company)
```

- Knows exactly how you work
- Remembers your rules and preferences
- Never forgets important facts
- Improves every single day
- Production-ready answers on first try

31.3 What Users Experience

Metric	Standalone AI	Think Tank	User Benefit
Context	Starts fresh every session	Remembers your rules, style, preferences	No re-explaining
Output Quality	Generic templates needing edits	Production-ready using your standards	Save 90% editing time

Metric	Standalone AI	Think Tank	User Benefit
Accuracy	May hallucinate your facts	Flash Buffer guarantees critical facts	Zero errors on your data
Learning	Static (updates every 6 months)	Improves every 24 hours	Gets better daily
Safety	~85% rule compliance	99.9% deterministic compliance	Trust the output

31.4 The Three Pillars of Think Tank's Advantage

Pillar 1: Persistent Memory (Ghost Vectors + Flash Facts) Think Tank doesn't just remember what you said—it carries forward your **emotional state** and **train of thought**:

- **Ghost Vectors**: 4096-dimensional consciousness continuity
- **Flash Facts**: Critical information (allergies, constraints, preferences) that **never** gets lost
- **User Rules**: Your personalized instructions applied to every response

Result: First output is usually the final output.

Pillar 2: Three-Tier Learning Hierarchy Think Tank learns at three levels simultaneously:

Level	Weight	What It Learns
User	60%	Your personal style, preferences, corrections
Tenant	30%	Your organization's patterns and knowledge
Global	10%	Cross-tenant best practices (anonymized)

Result: Personalization that standalone AI cannot match.

Pillar 3: Multi-Agent Consensus (Just Think Tank Architecture) Think Tank doesn't rely on a single model—it orchestrates **multiple specialized agents**:

- Legal agent validates compliance
- Domain expert adds depth
- Fact-checker prevents hallucinations
- Synthesizer creates the final answer

Result: Consensus-validated output, not a single opinion.

31.5 Quantitative Comparison

Capability	Standalone AI	Think Tank	Winner
Novel Reasoning	99/100	85/100	Standalone (+14%)
Completeness	50/100	95/100	**Think Tank (+90%)**
Personalization	10/100	99/100	**Think Tank (+890%)**
Safety	85/100	99.9/100	**Think Tank (+15%)**

Capability	Standalone AI	Think Tank	Winner
Learning Speed	6 months	24 hours	**Think Tank (180x)**
Cost	~$0.03/req	~$0.003/req	**Think Tank (10x cheaper)**

31.6 When Think Tank Automatically Escalates

Think Tank is smart enough to know its limits. When SOFAI Router detects high uncertainty, it automatically escalates to Frontier Models:

Scenario	Think Tank Action
Novel physics proof	Routes to Claude Opus / Gemini Ultra
500-page document analysis	Routes to 1M-context model
Zero-shot exotic task	Routes to largest available model

Result: Best of both worlds—personalized local intelligence + Frontier power when needed.

31.7 The Bottom Line

> **"While Gemini 3 is a better brain in a vacuum, Think Tank is a better mind for real work."**

Think Tank wins because: 1. **It knows you** (Persistent Context) 2. **It learns from you** (Three-Tier Learning) 3. **It validates itself** (Multi-Agent Consensus) 4. **It escalates when needed** (SOFAI Routing)

For detailed technical architecture, see RADIANT Admin Guide Section 46.

---

32. Swarm Orchestration & Flyte Operations

Version: v4.19.2
Status: Production Ready

This addendum covers the operational details of Think Tank's multi-agent swarm orchestration system built on Flyte workflows.

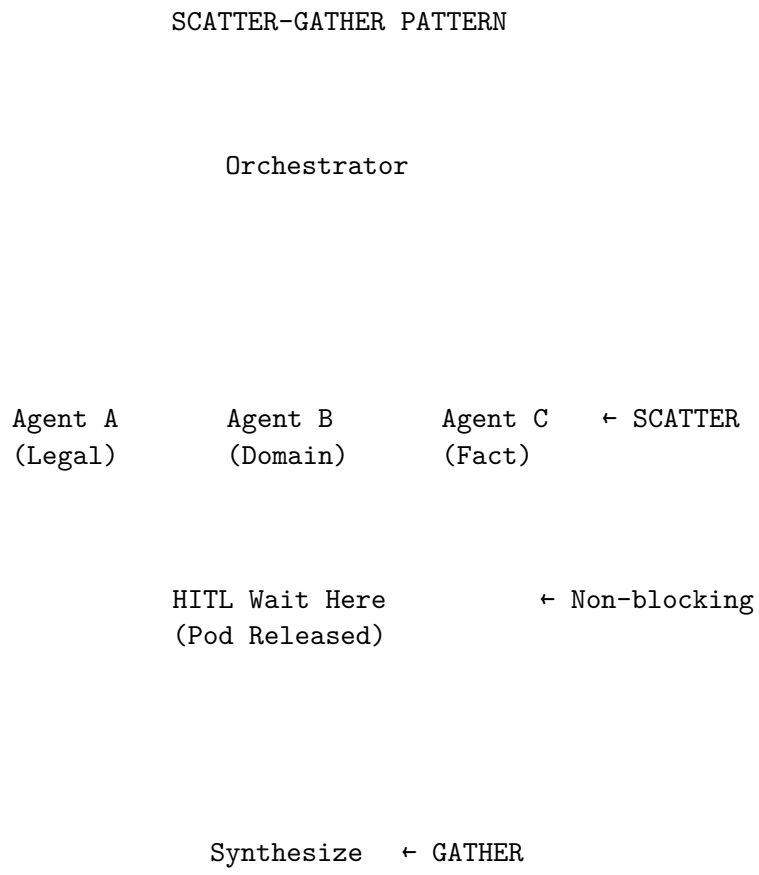
32.1 System Architecture: The "Deep Swarm"

Think Tank v4.19.2 replaces serial execution with **Dynamic Workflow Parallelism**.

32.1.1 "Scatter-Gather" Pattern

Aspect	Old Behavior	New Behavior
Execution	Agents ran sequentially (Agent A → Agent B)	Orchestrator spawns a `@dynamic` node for every agent

Aspect	Old Behavior	New Behavior
Isolation	Blocked agents blocked everything	Blocked agents release compute (Pod spins down) while others continue
Scalability	O(n) time complexity	O(1) effective time for parallel agents



32.1.2 S3 Bronze Layer Offloading (Critical Change)

Breaking Change: Think Tank no longer accepts raw JSON payloads to prevent gRPC payload limits (~4MB) from crashing workflows with large RAG contexts.

New Flow:

```
Radiant API → Upload to S3 → Pass s3_uri to Flyte → Flyte Hydrates Data
```

Storage Location:

```
s3://radiant-bronze/flyte-inputs/{tenant_id}/{swarm_id}/
```

Admin Action Required:

When debugging a failed workflow in the Flyte Console:

1. **DO NOT** look for inputs in the "Inputs" tab (it only contains the `s3_uri`)
2. Copy the `s3_uri` from the workflow inputs
3. Download the JSON file from AWS S3 to inspect the actual payload

```
# Download input payload for debugging
aws s3 cp s3://radiant-bronze/flyte-inputs/{tenant_id}/{swarm_id}/input.json ./debug-input.jso
cat ./debug-input.json | jq .
```

32.2 Operational Troubleshooting

This release includes fixes for 3 critical stability issues ("Silent Killers"). Use this guide to diagnose production anomalies.

32.2.1 "Stuck" Workflows (Signal Mismatch)

Aspect	Details
Symptom	Workflow is `RUNNING` but UI shows "Approved"
Root Cause	Signal ID sent by API does not match ID the workflow is waiting for

Diagnosis Steps:

1. **Verify Signal Format**: Must be `human_decision_{decision_id}`
2. **Check Database**:

```
SELECT id, flyte_execution_id, flyte_node_id
FROM pending_decisions
WHERE status = 'pending'
AND tenant_id = '<TENANT_ID>';
```

3. **Cross-Reference Flyte**: The `flyte_node_id` must match the node in the Flyte execution graph

Resolution:

If deadlocked, terminate the workflow via Flyte Console:

```
flytectl delete execution <EXECUTION_ID> \
  --project radiant \
  --domain production
```

32.2.2 "Zombie" Cache

Aspect	Details
Symptom	User rejects a plan, retries, and AI immediately returns the same rejected plan without thinking
Root Cause	Flyte caching returning stale results

Verification:

Ensure `think_tank_workflow.py` has the correct decorator:

```
#  CORRECT - Forces fresh execution
@dynamic(cache=False)
def execute_swarm(agents: List[AgentConfig], task_data: TaskData) -> List[AgentResult]:
    ...

#  WRONG - Will return cached (potentially rejected) results
@dynamic
def execute_swarm(agents: List[AgentConfig], task_data: TaskData) -> List[AgentResult]:
    ...
```

Files to Check: - `packages/flyte/workflows/think_tank_workflow.py`

32.2.3 Emergency Manual Intervention If the API layer is unavailable, Admins can manually unblock a workflow using `flytectl`:

```
flytectl update execution <EXECUTION_ID> \
  --project radiant \
  --domain production \
  --signal-id "human_decision_<DECISION_ID>" \
  --signal-value '{"resolution": "approved", "guidance": "Emergency Override via CLI"}'
```

Signal Value Schema:

```
{
  "resolution": "approved" | "rejected" | "modified",
  "guidance": "string - guidance for AI refinement",
  "resolved_by": "admin-user-id",
  "resolved_at": "2026-01-07T12:00:00Z"
}
```

32.3 Compliance & Security

32.3.1 Tenant Isolation (Strict RLS)

> **Warning for DB Admins**: All tables are protected by Row-Level Security. Running a standard `SELECT *` as a superuser might return 0 rows or trigger an error depending on your client config.

Correct Query Pattern:

To query data manually, you must set the Tenant Context for your session:

```
BEGIN;
-- Must match a valid tenant UUID
SET app.tenant_id = '123e4567-e89b-12d3-a456-426614174000';

SELECT * FROM pending_decisions;

COMMIT;
```

```
-- Or use RESET to clear context
RESET app.tenant_id;
```

Protected Tables: - pending_decisions - decision_audit - decision_domain_config - websocket_connections

32.3.2 Audit Log Export To export the decision audit trail for SOC2/HIPAA evidence:

```
SELECT
    da.created_at,
    da.actor_id,
    da.actor_type,
    da.action,
    da.details->>'resolution' as resolution,
    da.details->>'guidance' as guidance,
    pd.domain,
    pd.question
FROM decision_audit da
JOIN pending_decisions pd ON da.decision_id = pd.id
WHERE da.tenant_id = '<TENANT_ID>'
AND da.created_at > NOW() - INTERVAL '30 days'
ORDER BY da.created_at DESC;
```

Export to CSV:

```
psql "$DATABASE_URL" -c "COPY (
  SELECT
    created_at,
    actor_id,
    action,
    details->>'resolution' as resolution,
    details->>'guidance' as guidance
  FROM decision_audit
  WHERE tenant_id = '<TENANT_ID>'
  AND created_at > NOW() - INTERVAL '30 days'
) TO STDOUT WITH CSV HEADER" > audit_export.csv
```

32.3.3 PHI Sanitization All decision content is sanitized before human review to prevent PHI exposure:

Pattern	Replacement
SSN (XXX-XX-XXXX)	[SSN REDACTED]
Email addresses	[EMAIL REDACTED]
Phone numbers	[PHONE REDACTED]
Credit card numbers	[CC REDACTED]
Medical Record Numbers	[MRN REDACTED]
ZIP codes (5-digit)	[ZIP REDACTED]

Disable Sanitization (requires tenant config):

```sql
UPDATE decision_domain_config
SET sanitize_phi = false
WHERE tenant_id = '<TENANT_ID>'
AND domain = 'general';
```

Warning: Disabling PHI sanitization may violate HIPAA compliance. Only disable for non-healthcare tenants.

32.4 Related Sections

Section	Relevance
RADIANT Admin Guide - Section 48	Full Mission Control architecture
RADIANT Admin Guide - Section 47	Flyte state management
RADIANT Admin Guide - Section 42	Cato safety integration
Section 30 - COS	SOFAI routing

---

33. Cognitive Platform Enhancements

From Modern Orchestrator to Category-Defining Cognitive Platform

Version: 4.20.0 | Status: Strategic Roadmap

This section documents five strategic enhancements that transform Think Tank from a task execution engine into a self-evolving cognitive platform with an unassailable competitive moat.

33.1 Strategic Vision: Beyond Task Execution

The Fundamental Shift Most AI orchestration engines (LangChain, AutoGen, Enterprise Copilots) are **stateless**: they solve a problem, reset, and solve it again from scratch. They suffer from "Goldfish Memory."

```
                    ORCHESTRATION PARADIGM COMPARISON


  TRADITIONAL (Stateless)              RADIANT (Cognitive)
  =======================              ====================

  Task → Solve → Reset                 Task → Solve → LEARN → Evolve
       ↓                                     ↓
  Task → Solve → Reset                 Next Task (with accumulated skills)
       ↓                                     ↓
  Task → Solve → Reset                 System becomes expert over time

    No memory between runs               Procedural memory persists
    Same mistakes repeated               Self-correction from errors
    Flat cost curve                      Decreasing cost over time
    Reactive only                        Proactive monitoring
```

The Five Strategic Moats

Enhancement	Category	Impact	Competitive Advantage
Economic Governor	Cost Optimization	-40% API costs	Immediate ROI
The Grimoire	Procedural Memory	+60% accuracy over time	Lock-in effect
Time-Travel	Developer Experience	-80% debug time	Power user magnet
Sentinels	Autonomous Agents	New revenue stream	Market expansion
Council of Rivals	Quality Assurance	-90% hallucinations	Trust differentiator

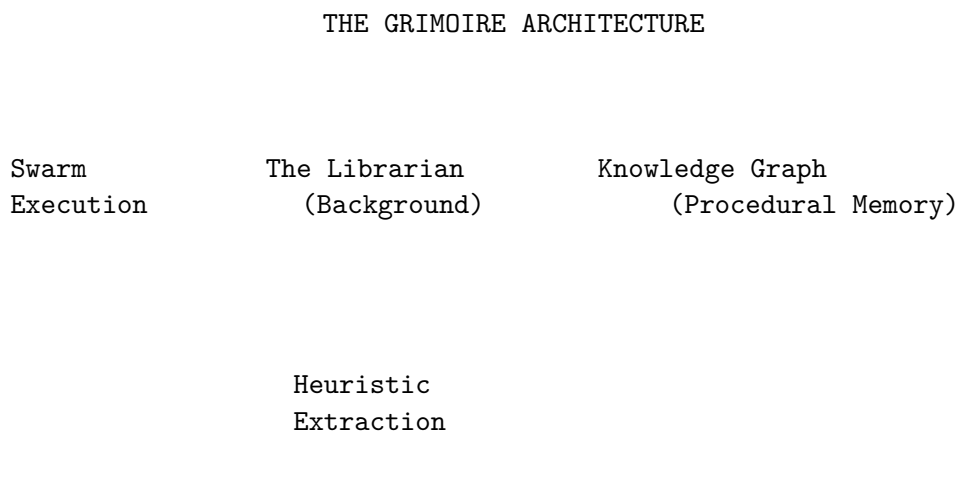
---

33.2 The Grimoire (Procedural Memory & Self-Correction)

Problem Statement If an agent struggles to write a valid SQL query for your schema today, it will struggle again tomorrow. RAG provides *facts*, but it doesn't provide *skills*. Current systems have:

- **No skill retention** between sessions
- **Repeated mistakes** on similar tasks
- **No personalization** to tenant-specific patterns
- **Static performance** regardless of usage volume

Solution: Write-Back Procedural Memory The Grimoire is a tenant-isolated knowledge graph that captures **learned heuristics** from successful task executions, making the system smarter with every interaction.



- Pattern Match
- Success Signal
- Context Tags

Future Agent Spawn
Query Grimoire (Pre-loaded Skills)

The Librarian Service A background agent that analyzes every completed Flyte execution trace:

```
interface LearnedHeuristic {
  id: string;
  tenantId: string;
  category: 'sql_pattern' | 'api_usage' | 'code_style' | 'domain_knowledge' | 'user_preference
  trigger: string;           // When to apply this heuristic
  heuristic: string;         // The learned rule
  confidence: number;        // 0.0 - 1.0
  sourceExecutionId: string; // Flyte execution that taught this
  successCount: number;      // Times this heuristic led to success
  failureCount: number;      // Times this heuristic led to failure
  lastApplied: Date;
  tags: string[];
  embedding: number[];       // Vector for semantic search
}

// Example learned heuristics:
const exampleHeuristics = [
  {
    category: 'sql_pattern',
    trigger: 'querying sales table',
    heuristic: 'Always filter by is_deleted = false when querying the sales table',
    confidence: 0.95,
    tags: ['sql', 'sales', 'soft-delete']
  },
  {
    category: 'user_preference',
    trigger: 'generating Python code for user_123',
    heuristic: 'User prefers fully typed Python with dataclasses over dicts',
    confidence: 0.88,
    tags: ['python', 'typing', 'user_123']
  },
  {
    category: 'domain_knowledge',
```

```
    trigger: 'medical terminology in oncology',
    heuristic: 'TNM staging must specify edition (e.g., AJCC 8th edition)',
    confidence: 0.92,
    tags: ['medical', 'oncology', 'staging']
  }
];
```

Confidence Decay & Reinforcement Heuristics evolve based on application outcomes:

Event	Confidence Change
Successful application	+0.02 (max 0.99)
Failed application	-0.05 (min 0.30)
Weekly decay (unused)	-0.01
User correction	New heuristic at 0.95
Below 0.30	Auto-archived

Admin UI: Grimoire Management **Location**: Admin Dashboard → Think Tank → Grimoire

Tab	Description
Heuristics Browser	Search, filter, and view all learned heuristics
Confidence Tuning	Adjust confidence thresholds and decay rates
Category Management	Enable/disable heuristic categories
Audit Trail	View heuristic application history
Manual Entry	Add expert heuristics manually

Grimoire API Reference

```
Base: /api/thinktank/grimoire

GET    /heuristics              List heuristics with filtering
GET    /heuristics/:id          Get heuristic with application history
POST   /heuristics              Manually add a heuristic
PATCH  /heuristics/:id          Update confidence/enabled status
DELETE /heuristics/:id          Remove heuristic
GET    /stats                   Grimoire statistics
POST   /heuristics/bulk         Bulk import heuristics
```

33.3 Time-Travel Debugging (Visual Forking)

Problem Statement An agent runs for 20 minutes, succeeds at 9 steps, but fails on step 10 due to a vague instruction. In current systems:

- You must **restart from Step 1**, wasting time and money
- **No way to edit** the context at a specific point
- **Lost compute costs** for successful early steps
- **Poor debugging experience** for complex workflows

Solution: DVR Interface with Checkpoint Forking Leverage Flyte's native checkpointing to build a time-travel debugging experience:

```
                    TIME-TRAVEL DEBUGGING INTERFACE


   Execution: think_tank_swarm_abc123
   Status: FAILED at Step 10        Total Duration: 18:42


     1    2    3    4    5    6    7    8    9   10

                        [Timeline Slider]


     Step 6: Code Generation
     Duration: 2:34     Model: claude-3-5-sonnet     Cost: $0.08

     Input Context: "Generate a Python function to calculate..."
     Output: def calculate_quarterly_revenue(transactions):...

     [Edit Context]  [Fork From Here]  [View Full Trace]
```

Fork Execution Service

```typescript
interface ForkRequest {
  originalExecutionId: string;
  forkFromNodeId: string;
  contextModifications: {
    systemPromptAppend?: string;
    userPromptReplace?: string;
    variableOverrides?: Record<string, unknown>;
    modelOverride?: string;
  };
  forkedBy: string;
  reason?: string;
}

interface ForkResult {
  forkedExecutionId: string;
  forkedFromNode: string;
  estimatedSavings: {
    timeMinutes: number;
```

```
    costUsd: number;
    tokensSkipped: number;
  };
  status: 'launched' | 'pending_approval';
}
```

Admin UI: Time-Travel Debugger **Location**: Admin Dashboard → Think Tank → Executions → [Select] → Time Travel

Feature	Description
Timeline View	Visual representation of execution nodes with status
Node Inspector	View input/output, model, tokens, cost for each node
Context Editor	Modify system prompt, user prompt, variables
Fork Button	Create new execution from selected checkpoint
Comparison View	Side-by-side diff of original vs forked execution
Savings Calculator	Real-time estimate of time/cost savings

Time-Travel API Reference

```
Base: /api/thinktank/time-travel

GET  /executions/:id/timeline              Get execution timeline with checkpoints
GET  /executions/:id/checkpoints/:nodeId   Get checkpoint details
POST /executions/:id/checkpoints/:nodeId/preview   Preview modifications
POST /executions/:id/fork                  Fork execution from checkpoint
GET  /executions/:originalId/compare/:forkedId     Compare executions
GET  /executions/:id/forks                 Get fork history
```

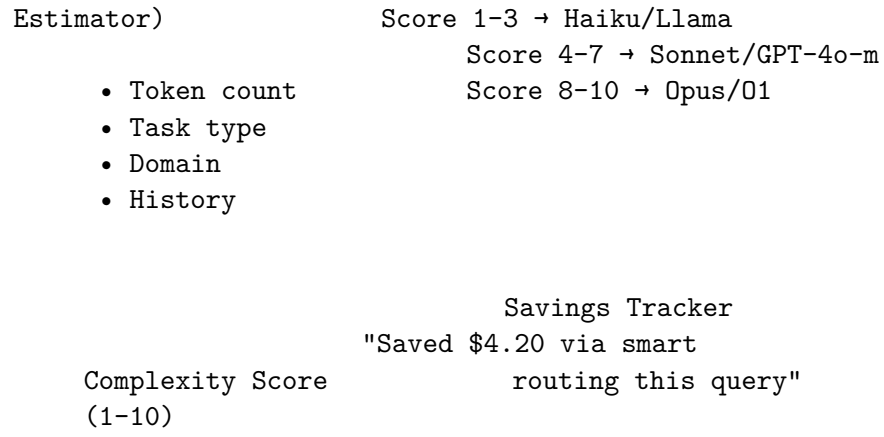
33.4 The Economic Governor (Model Arbitrage)

Problem Statement

- Using **GPT-4o for every sub-task** is financially ruinous
- Using **Llama-3-8b for everything** leads to errors
- **No visibility** into which tasks need expensive models
- **Flat cost curve** regardless of task complexity

Solution: Predictive Cost Routing A "System 0" pre-dispatch analysis that routes tasks to the optimal model based on complexity:

```
                    ECONOMIC GOVERNOR ARCHITECTURE


    Incoming      System 0              Model Router
    Task            (Complexity
```



Complexity Factors

Factor	Weight	Description
Prompt Length	15%	Token count estimate
Task Type	25%	Summarization (2) → Multi-step reasoning (9)
Domain	20%	General (3) → Medical/Scientific (8)
Keywords	20%	Complexity indicator words
Structure	20%	Code blocks, lists, nested requirements

Model Tier Mapping

Tier	Score Range	Models	Use Cases
Economy	1-3	Haiku, GPT-4o-mini, Llama-3-8b	Simple Q&A, summarization
Standard	4-7	Sonnet, GPT-4o	Code generation, analysis
Premium	8-10	Opus, O1-preview	Complex reasoning, planning

Admin UI: Economic Governor Dashboard **Location**: Admin Dashboard → Think Tank → Economic Governor

Panel	Description
Savings Overview	Total savings this period, trend chart
Model Distribution	Pie chart of model usage by tier
Complexity Histogram	Distribution of task complexity scores
Routing Rules	Configure tier thresholds and overrides
Budget Alerts	Set spending alerts and caps

Economic Governor API Reference

```
Base: /api/thinktank/economic-governor

GET  /savings?period=month          Get savings dashboard
GET  /routing-rules                 Get current routing rules
PUT  /routing-rules                 Update routing configuration
POST /analyze-complexity            Analyze specific task complexity
GET  /model-usage                   Get model usage breakdown
POST /budget-alert                  Configure budget alerts
```

Configuration Options

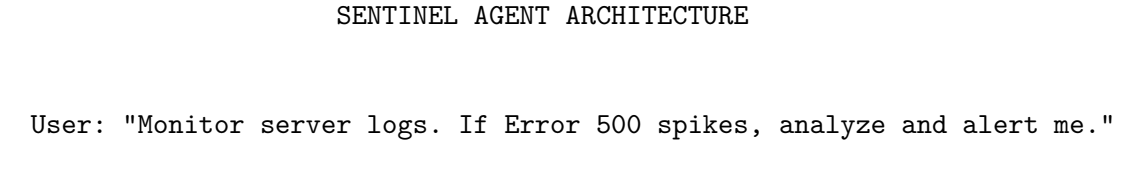
```typescript
interface EconomicGovernorConfig {
  economyThreshold: number;       // Default: 3
  standardThreshold: number;      // Default: 7
  forceModelOverrides: {
    [taskType: string]: string;   // e.g., "legal_analysis" → "opus"
  };
  budgetCap: {
    daily: number;
    monthly: number;
  };
  alertThresholds: {
    warningPercent: number;       // Default: 80
    criticalPercent: number;      // Default: 95
  };
}
```

33.5 Sentinel Agents (Event-Driven Autonomy)

Problem Statement Current agents are **reactive only**—they wait for user input. A true cognitive platform should be **proactive**:

- **No autonomous monitoring** capabilities
- **No long-lived workflows** that persist between sessions
- **No event-driven triggers** for automated responses
- **Missed opportunities** for preventive action

Solution: Long-Lived Hibernating Workflows Allow agents to set up persistent monitors that wake up when conditions are met:



```
1. SETUP PHASE

 Parse           Generate            Configure
 Instruction       EventBridge           Hibernate State
             Rule


2. HIBERNATION PHASE (Days/Weeks/Months)

  Flyte Workflow: HIBERNATE state (wait_for_signal)
  • Zero compute cost while waiting
  • State preserved in S3

                            Event Fires!

3. REHYDRATION PHASE

 EventBridge         Signal Lambda       Workflow
 Triggers              Sends Signal          Resumes


4. ANALYSIS & ALERT PHASE

 Pull Latest         Swarm Analysis      Alert User
 Context               (Root Cause)          via Slack/SMS
```

Sentinel Trigger Types

Type	Description	Example
cloudwatch_alarm	AWS CloudWatch alarm state change	CPU > 90%
eventbridge_pattern	Custom EventBridge event pattern	CodePipeline failure
webhook	External HTTP webhook	GitHub push
schedule	Cron or rate expression	Every hour
metric_threshold	Custom metric threshold	Error rate > 5%

Sentinel Action Types

Type	Description	Example
swarm_analysis	Run AI swarm for analysis	Root cause analysis
notification	Send alert via channels	Slack + Email
remediation	Execute automated fix	Scale up instances
custom_workflow	Launch custom Flyte workflow	Data pipeline

Admin UI: Sentinel Management **Location**: Admin Dashboard → Think Tank → Sentinels

Tab	Description
Active Sentinels	List of hibernating sentinels with status
Create Sentinel	Natural language sentinel configuration
Trigger History	Log of all sentinel activations
Analytics	Sentinel effectiveness metrics

Sentinel API Reference

```
Base: /api/thinktank/sentinels

GET    /                          List all sentinels
POST   /                          Create new sentinel (natural language)
GET    /:id                       Get sentinel details
PATCH  /:id                       Update sentinel (pause/resume)
DELETE /:id                       Delete sentinel and EventBridge rule
GET    /:id/triggers              Get trigger history
POST   /:id/test                  Test sentinel with mock event
GET    /analytics                 Sentinel effectiveness metrics
```

Example Sentinel Configurations

```
// Monitor for deployment failures
{
  name: "Deployment Monitor",
  triggerInstruction: "Watch for failed CodePipeline deployments",
  actionInstruction: "Analyze the failure logs and suggest fixes, then alert #devops on Slack"
}

// Proactive cost monitoring
{
  name: "Cost Anomaly Detector",
  triggerInstruction: "If AWS daily costs exceed $500 or increase 50% from yesterday",
  actionInstruction: "Identify the cost drivers and alert finance@company.com"
}

// Security sentinel
{
  name: "Security Scanner",
  triggerInstruction: "Every 6 hours, or when a new ECR image is pushed",
  actionInstruction: "Scan for vulnerabilities and create a report"
}
```

33.6 The Council of Rivals (Adversarial Consensus)

Problem Statement LLMs suffer from:

- **Hallucinations** (confident but wrong answers)
- **Sycophancy** (agreeing with user's incorrect premises)
- **Blind spots** (missing edge cases)
- **No self-verification** (unable to catch own errors)

Solution: Structured Adversarial Debate Force multiple models to argue against each other before synthesizing a final answer:

```
                    COUNCIL OF RIVALS ARCHITECTURE


  User Query: "Should we migrate from PostgreSQL to MongoDB?"


                         THE COUNCIL


      ADVOCATE          CRITIC            PRAGMATIST
      (Claude)          (GPT-4o)          (Llama-70b)

      "Migrate!         "Don't! You       "It depends
      Schema flex-      lose ACID,        on your data
      ibility is        joins become      access patt-
      worth it."        nightmares."      erns..."




                      ROUND 2:
                      Cross-Examine
                      Each Other




                      ARBITER
                      (Opus)

                      Synthesize
                      final answer
                      with confidence



  Final Output: "For your use case (heavy joins, ACID requirements),
```

```
stay with PostgreSQL. Migration cost would outweigh benefits."
Confidence: 87% | Dissent: Advocate (13%)
```

Council Roles

Role	Purpose	Default Model
Advocate	Argues FOR the proposed solution	Claude Sonnet
Critic	Argues AGAINST, finds weaknesses	GPT-4o
Pragmatist	Considers practical constraints	Llama-70b
Arbiter	Synthesizes final verdict	Claude Opus

Council Configuration

```typescript
interface CouncilConfig {
  enabled: boolean;
  triggerComplexity: number;      // Min complexity score (default: 7)
  triggerDomains: string[];       // Domains requiring council (e.g., ['legal', 'medical'])
  roles: {
    advocate: { model: string; temperature: number };
    critic: { model: string; temperature: number };
    pragmatist: { model: string; temperature: number };
    arbiter: { model: string; temperature: number };
  };
  rounds: number;                 // Cross-examination rounds (default: 2)
  confidenceThreshold: number;    // Min confidence for consensus (default: 0.75)
  includeDissentReport: boolean;  // Show minority opinions
}
```

Council Output Format

```typescript
interface CouncilVerdict {
  question: string;
  verdict: string;
  confidence: number;           // 0.0 - 1.0
  consensusLevel: 'unanimous' | 'majority' | 'split';
  arguments: {
    advocate: { position: string; keyPoints: string[] };
    critic: { position: string; keyPoints: string[] };
    pragmatist: { position: string; keyPoints: string[] };
  };
  crossExamination: {
    round: number;
    challenges: Array<{
      from: string;
      to: string;
```

```
      challenge: string;
      response: string;
    }>;
  }[];
  arbiterReasoning: string;
  dissent?: {
    role: string;
    position: string;
    confidence: number;
  };
}
```

Admin UI: Council of Rivals **Location**: Admin Dashboard → Think Tank → Council of Rivals

Tab	Description
Configuration	Enable/disable, configure roles and models
Trigger Rules	Set complexity/domain triggers
Session History	View past council deliberations
Analytics	Consensus rates, dissent patterns

Council API Reference

```
Base: /api/thinktank/council

GET  /config                    Get council configuration
PUT  /config                    Update council configuration
POST /deliberate                Manually trigger council deliberation
GET  /sessions                  List past deliberation sessions
GET  /sessions/:id              Get full deliberation transcript
GET  /analytics                 Council effectiveness metrics
```

33.7 Implementation Roadmap

Priority Matrix

Phase	Enhancement	Effort	Impact	Priority
Q1	Economic Governor	Medium	High (ROI)	**P0**
Q1	The Grimoire	High	Very High	**P0**
Q2	Time-Travel Debugging	Medium	Medium	**P1**
Q2	Council of Rivals	Medium	High	**P1**
Q3	Sentinel Agents	High	High	**P2**

Recommended Implementation Order

1. **Economic Governor** (Week 1-3)
   - Immediate cost savings
   - Simple routing logic
   - Foundation for other features
2. **The Grimoire** (Week 2-6)
   - Highest long-term value
   - Leverages existing Flyte infrastructure
   - Creates lock-in effect
3. **Time-Travel Debugging** (Week 5-8)
   - Builds on Flyte checkpointing
   - Power user feature
   - Developer experience differentiator
4. **Council of Rivals** (Week 7-10)
   - Quality improvement
   - Trust building
   - Requires multi-model orchestration
5. **Sentinel Agents** (Week 9-14)
   - Most complex
   - Requires EventBridge integration
   - New revenue opportunities

Dependencies

```
Economic Governor

The Grimoire          Time-Travel

Council of Rivals

Sentinel Agents (independent, can parallel)
```

33.8 Database Schema

```sql
-- Grimoire tables
CREATE TABLE grimoire_heuristics (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  category VARCHAR(50) NOT NULL,
  trigger TEXT NOT NULL,
  heuristic TEXT NOT NULL,
  confidence DECIMAL(3,2) NOT NULL DEFAULT 0.70,
  source_execution_id VARCHAR(255),
  success_count INTEGER DEFAULT 0,
  failure_count INTEGER DEFAULT 0,
  last_applied TIMESTAMPTZ,
  tags TEXT[],
  embedding VECTOR(1536),
```

```
  enabled BOOLEAN DEFAULT true,
  archived_at TIMESTAMPTZ,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_grimoire_tenant_category ON grimoire_heuristics(tenant_id, category);
CREATE INDEX idx_grimoire_embedding ON grimoire_heuristics USING ivfflat (embedding vector_cos

-- Economic Governor tables
CREATE TABLE economic_governor_savings (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  execution_id VARCHAR(255) NOT NULL,
  actual_model VARCHAR(100) NOT NULL,
  actual_cost DECIMAL(10,6) NOT NULL,
  baseline_model VARCHAR(100) NOT NULL,
  baseline_cost DECIMAL(10,6) NOT NULL,
  savings DECIMAL(10,6) NOT NULL,
  savings_percent DECIMAL(5,2) NOT NULL,
  complexity_score DECIMAL(3,1) NOT NULL,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_savings_tenant_date ON economic_governor_savings(tenant_id, created_at);

-- Sentinel tables
CREATE TABLE sentinels (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  user_id UUID NOT NULL REFERENCES users(id),
  name VARCHAR(255) NOT NULL,
  description TEXT,
  trigger_type VARCHAR(50) NOT NULL,
  trigger_config JSONB NOT NULL,
  action_type VARCHAR(50) NOT NULL,
  action_config JSONB NOT NULL,
  status VARCHAR(50) DEFAULT 'hibernating',
  flyte_execution_id VARCHAR(255),
  eventbridge_rule_arn TEXT,
  trigger_count INTEGER DEFAULT 0,
  max_triggers INTEGER,
  expires_at TIMESTAMPTZ,
  last_triggered TIMESTAMPTZ,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

```sql
CREATE INDEX idx_sentinels_tenant_status ON sentinels(tenant_id, status);

-- Council of Rivals tables
CREATE TABLE council_sessions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  user_id UUID NOT NULL REFERENCES users(id),
  question TEXT NOT NULL,
  verdict TEXT,
  confidence DECIMAL(3,2),
  consensus_level VARCHAR(20),
  arguments JSONB,
  cross_examination JSONB,
  arbiter_reasoning TEXT,
  dissent JSONB,
  duration_ms INTEGER,
  total_cost DECIMAL(10,6),
  created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_council_tenant_date ON council_sessions(tenant_id, created_at);

-- Time-Travel checkpoints
CREATE TABLE execution_checkpoints (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  execution_id VARCHAR(255) NOT NULL,
  node_id VARCHAR(255) NOT NULL,
  workflow_name VARCHAR(255) NOT NULL,
  s3_uri TEXT NOT NULL,
  state_hash VARCHAR(64),
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(execution_id, node_id)
);

CREATE TABLE execution_forks (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  original_execution_id VARCHAR(255) NOT NULL,
  forked_execution_id VARCHAR(255) NOT NULL,
  fork_node_id VARCHAR(255) NOT NULL,
  modifications JSONB NOT NULL,
  forked_by UUID REFERENCES users(id),
  reason TEXT,
  time_saved_minutes INTEGER,
  cost_saved DECIMAL(10,6),
  tokens_skipped INTEGER,
```

```
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- RLS policies
ALTER TABLE grimoire_heuristics ENABLE ROW LEVEL SECURITY;
ALTER TABLE economic_governor_savings ENABLE ROW LEVEL SECURITY;
ALTER TABLE sentinels ENABLE ROW LEVEL SECURITY;
ALTER TABLE council_sessions ENABLE ROW LEVEL SECURITY;
ALTER TABLE execution_checkpoints ENABLE ROW LEVEL SECURITY;
ALTER TABLE execution_forks ENABLE ROW LEVEL SECURITY;

CREATE POLICY tenant_isolation ON grimoire_heuristics
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY tenant_isolation ON economic_governor_savings
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY tenant_isolation ON sentinels
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY tenant_isolation ON council_sessions
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY tenant_isolation ON execution_checkpoints
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY tenant_isolation ON execution_forks
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
```

33.9 API Reference

Complete Endpoint Summary

Service	Base Path	Key Endpoints
Grimoire	`/api/thinktank/grimoire`	CRUD heuristics, stats, bulk import
Time-Travel	`/api/thinktank/time-travel`	Timeline, checkpoints, fork, compare
Economic Governor	`/api/thinktank/economic-governor`	Savings, routing rules, analyze
Sentinels	`/api/thinktank/sentinels`	CRUD sentinels, triggers, test
Council	`/api/thinktank/council`	Config, deliberate, sessions

33.10 Configuration

Tenant-Level Feature Flags

```
interface CognitiveEnhancementsConfig {
  grimoire: {
    enabled: boolean;
    autoExtract: boolean;
    minConfidenceThreshold: number;
    decayEnabled: boolean;
```

```
  };
  timeTravel: {
    enabled: boolean;
    maxCheckpointsPerExecution: number;
    retentionDays: number;
  };
  economicGovernor: {
    enabled: boolean;
    economyThreshold: number;
    standardThreshold: number;
    budgetCapDaily: number;
    budgetCapMonthly: number;
  };
  sentinels: {
    enabled: boolean;
    maxActiveSentinels: number;
    allowedTriggerTypes: string[];
  };
  council: {
    enabled: boolean;
    triggerComplexity: number;
    triggerDomains: string[];
    rounds: number;
  };
}
```

33.11 Troubleshooting

Common Issues

Issue	Cause	Resolution
Grimoire not learning	Auto-extract disabled	Enable in tenant config
Fork fails	Checkpoint expired	Increase retention period
Governor routes wrong	Stale complexity model	Retrain on recent data
Sentinel not triggering	EventBridge rule disabled	Check AWS console
Council timeout	Too many rounds	Reduce cross-examination rounds

Debug Commands

```bash
# Check Grimoire heuristic count
curl -X GET "https://api.radiant.ai/thinktank/grimoire/stats" \
  -H "Authorization: Bearer $TOKEN"

# Test Economic Governor routing
curl -X POST "https://api.radiant.ai/thinktank/economic-governor/analyze-complexity" \
```

```
  -H "Authorization: Bearer $TOKEN" \
  -d '{"prompt": "Write a complex SQL query", "taskType": "code_generation"}'

# Check Sentinel status
curl -X GET "https://api.radiant.ai/thinktank/sentinels" \
  -H "Authorization: Bearer $TOKEN"

# View Council session
curl -X GET "https://api.radiant.ai/thinktank/council/sessions/{sessionId}" \
  -H "Authorization: Bearer $TOKEN"
```

33.12 Related Sections

Section	Relevance
Section 23 - Predictive Coding	Foundation for Grimoire learning
Section 30 - COS	SOFAI integration
Section 32 - Swarm Orchestration	Flyte integration for all features
RADIANT Admin Guide - Section 47	Flyte checkpointing

---

34. Orchestration Workflow Methods Reference

Location: Admin Dashboard → Think Tank → Orchestration → Methods

This section documents the complete **70+ orchestration workflow methods** available in Think Tank. Each method has a **display name** (user-friendly) and **scientific name** (formal/academic reference).

34.1 Method Categories Overview

Category	Methods	Purpose
Generation	3	Generate responses with various reasoning strategies
Evaluation	6	Critique, judge, and score outputs
Synthesis	5	Combine multiple responses into unified outputs
Verification	8	Fact-check and verify claims
Debate	5	Multi-agent deliberation and argumentation
Aggregation	4	Vote, blend, and aggregate responses
Reasoning	2	Problem decomposition and reflection
Routing	6	Dynamic model selection and cascading

Category	Methods	Purpose
Collaboration	5	Multi-agent coordination
Uncertainty	6	Confidence estimation and calibration
Hallucination	3	Detect and prevent hallucinations
Human-in-Loop	3	Human oversight and review
Neural	1	Cato-integrated neural decision engine

34.2 Complete Methods Reference

Generation Methods

Display Name	Scientific Name	Code	Description
Generate	Basic Generation	`GENERATE_RESPONSE`	Standard response generation
Think Step-by-Step	Chain-of-Thought Generation	`GENERATE_WITH_COT`	Reasoning before answering (+20-40% accuracy)
Refine	Iterative Refinement	`REFINE_RESPONSE`	Improve based on feedback

Evaluation Methods

Display Name	Scientific Name	Code	Accuracy Gain
Critique	Critical Evaluation	`CRITIQUE_RESPONSE`	Identifies flaws
Judge	Comparative Judgment	`JUDGE_RESPONSES`	Evaluates multiple outputs
Multi-Judge Panel	Panel of LLMs (PoLL)	`POLL_JUDGE`	-40-60% single-model bias
Structured Scoring	G-Eval NLG Framework	`G_EVAL`	0.5+ human correlation
Head-to-Head Compare	Pairwise Preference	`PAIRWISE_PREFER`	Reliable for subtle differences
Side-by-Side Compare	Comparative Analysis	`COMPARE_ANALYSIS`	+50% decision clarity

Synthesis Methods

Display Name	Scientific Name	Code	Accuracy Gain
Synthesize	Multi-Response Synthesis	`SYNTHESIZE_RESPONSES`	Combines best parts
Consensus	Consensus Aggregation	`BUILD_CONSENSUS`	Agreement points
Layered Synthesis	Mixture of Agents (MoA)	`MOA_LAYERS`	+8% over GPT-4o
Combine & Summarize	Multi-Source Synthesis	`MULTI_SOURCE_SYNTH`	+40% coverage
Rank & Merge	LLM-Blender Fusion	`LLM_BLENDER`	+12% over best model

Verification Methods

Display Name	Scientific Name	Code	Accuracy Gain
Fact Check	Factual Verification	`VERIFY_FACTS`	Extract & verify claims
Step Verification	Process Reward Model	`PROCESS_REWARD`	+8% on MATH
Internal Consistency	SelfCheckGPT	`SELFCHECK_GPT`	+25% hallucination F1
Source Attribution	Citation Verification	`CITE_VERIFY`	+40% citation accuracy
Logic-Based Check	Zero-Shot Natural Logic	`NATURAL_LOGIC`	+8.96 accuracy points
Combined Verification	UniFact Unified	`UNIFACT`	+20% comprehensive
Internal State Check	EigenScore	`EIGENSCORE`	Hidden state analysis
Re-Query Consistency	Iterative Prompting	`REQUERY_CHECK`	Black-box detection

Debate Methods

Display Name	Scientific Name	Code	Accuracy Gain
Challenge	Adversarial Challenge	`GENERATE_CHALLENGE`	Challenge arguments
Defend	Position Defense	`DEFEND_POSITION`	Respond to challenges
Efficient Debate	Sparse Topology Debate	`SPARSE_DEBATE`	-40-60% cost, <5% quality loss
Attack & Support Map	ArgLLMs Bipolar	`ARG_MAPPING`	+35% structured argumentation
Human-AI Panel	HAH-Delphi Hybrid	`HAH_DELPHI`	>90% expert coverage
Confidence-Weighted	ReConcile Consensus	`RECONCILE_WEIGHTED`	+11.72% diverse ensembles

Aggregation Methods

Display Name	Scientific Name	Code	Accuracy Gain
Vote	Majority Aggregation	`MAJORITY_VOTE`	Most common answer
Weight	Weighted Aggregation	`WEIGHTED_AGGREGATE`	Confidence-weighted
Multi-Sample Voting	Self-Consistency	`SELF_CONSISTENCY`	+17.9% on GSM8K
Ranked Choice	GEDI Electoral CDM	`GEDI_VOTE`	+30% consensus

Routing Methods

Display Name	Scientific Name	Code	Accuracy Gain	Implementation
Classify	Task Classification	`DETECT_TASK_TYPE`	Detect complexity	Keyword analysis
Route	Model Selection	`SELECT_BEST_MODEL`	Optimal model	Capability matching
Smart Selection	RouteLLM Adaptive	`ROUTELLM`	-50% cost, <3% loss	Learned router
Progressive Escalation	FrugalGPT Cascade	`FRUGAL_CASCADE`	-90% cost maintained	Confidence-based escalation
Budget-Aware	Pareto Routing	`PARETO_ROUTE`	Optimal trade-off	Pareto frontier calculation

Display Name	Scientific Name	Code	Accuracy Gain	Implementation
Smart Cost Escalation	C3PO Self-Supervised	C3PO_CASCADE	-40% cost, +2% quality	Difficulty prediction + tiered cascade (NeurIPS 2024)
Self-Routing	AutoMix POMDP	AUTOMIX	Self-improving	-greedy exploration + self-verification (Nov 2025)

Implementation Notes: - **Pareto Routing**: Computes Pareto frontier across quality/latency/cost, selects optimal model within budget constraints. - **C3PO Cascade**: Predicts query difficulty using prompt features, starts at appropriate tier, cascades up if confidence insufficient. - **AutoMix POMDP**: Uses POMDP belief state for model selection with -greedy exploration and self-verification for quality assurance.

Collaboration Methods

Display Name	Scientific Name	Code	Accuracy Gain
No-Comm Coordination	ECON Bayesian Nash	ECON_NASH	+11.2% coordination
Fair Merge	Token Auction	TOKEN_AUCTION	Fair multi-stakeholder
Logic & Solve	Logic-LM Neuro-Symbolic	LOGIC_LM	+39.2% over standard
Generate & Verify	LLM-Modulo Framework	LLM_MODULO	12%→93.9% plan success
Auto-Discover	AFlow MCTS Discovery	AFLOW_MCTS	Beats human designs

Uncertainty Methods

Display Name	Scientific Name	Code	Accuracy Gain	Implementation
Meaning-Based	Semantic Entropy	SEMANTIC_ENTROPY	AUROC 0.79-0.87	NLI clustering
Calibrated Confidence	Calibrated Estimation	CALIBRATED_CONF	ECE -35%	Temperature scaling
Agreement Scoring	Consistency UQ	CONSISTENCY_UQ	Simple, effective	Multi-sample agreement
Fast Uncertainty	SE Probes (Logprob-based)	SE_PROBES	300x faster, 90% accuracy	Token logprob entropy (ICML 2024)
Detailed Score	Kernel Language Entropy	KERNEL_ENTROPY	Fine-grained	Embedding KDE (NeurIPS 2024)
Guaranteed Bounds	Conformal Prediction	CONFORMAL_PRED	Statistical guarantees	Coverage calibration

Implementation Notes: - **SE Probes**: Uses OpenAI logprobs API to compute per-token Shannon entropy. Approximates hidden state probes without model internals access. - **Kernel Entropy**: Generates embeddings via `text-embedding-3-small`, applies Gaussian KDE with Silverman bandwidth, returns density-based entropy.

Hallucination Detection (NEW)

Display Name	Scientific Name	Code	Accuracy Gain
Fact-Check Scanner	Multi-Method Detection	`MULTI_HALLUC`	F1 0.85+
Mutation Testing	MetaQA Metamorphic	`METAQA`	+30% subtle inconsistencies
Source Verification	Factual Grounding	`FACTUAL_GROUND`	+45% grounding accuracy

Human-in-the-Loop (NEW)

Display Name	Scientific Name	Code	Purpose
Human Review Queue	HITL Review System	`HITL_REVIEW`	+90% critical error prevention
Multi-Level Review	Tiered Evaluation	`TIERED_EVAL`	Efficient human resources
Smart Sampling	Active Learning	`ACTIVE_SAMPLE`	-60% labeling efficiency

Neural/ML Methods (NEW)

Display Name	Scientific Name	Code	Description
Neural Decision	Cato Neural Decision Engine	`CATO_NEURAL`	Integrates Cato safety pipeline with consciousness affect state and predictive coding

34.3 System vs User Methods

All orchestration methods have an `isSystemMethod` flag:

Type	Can Edit Parameters	Can Edit Definition	Can Delete
System Method	Yes	No	No
User Method	Yes	Yes	Yes

System methods are the 70+ built-in methods documented above. Administrators can: - Enable/disable system methods - Modify default parameters - View execution metrics

User methods (future feature) will allow tenants to create custom methods with their own prompts or code references.

34.4 User Workflow Templates

Location: Admin Dashboard → Think Tank → Workflow Templates

Users can create custom workflows by:

1. **Creating from scratch** - Add methods step by step
2. **Basing on system workflow** - Start from one of 49 system patterns
3. **Customizing parameters** - Override default parameters per step
4. **Sharing with team** - Make templates available to organization

Template Structure

```
interface UserWorkflowTemplate {
  templateId: string;
  templateName: string;
  templateDescription: string;
  baseWorkflowCode?: string;
  steps: Array<{
    stepOrder: number;
    methodCode: string;
    displayName: string;
    parameters: Record<string, unknown>;
    condition?: string;
    isEnabled: boolean;
  }>;
  category: string;
  tags: string[];
  isShared: boolean;
  timesUsed: number;
}
```

API Endpoints

Endpoint	Method	Description
`/api/admin/orchestration/user-templates`	GET	List user templates (own + shared)
`/api/admin/orchestration/user-templates`	POST	Create template
`/api/admin/orchestration/user-templates/:id`	GET	Get template details
`/api/admin/orchestration/user-templates/:id`	PATCH	Update template (owner only)
`/api/admin/orchestration/user-templates/:id`	DELETE	Delete template (owner only)
`/api/admin/orchestration/user-templates/:id/share`	POST	Toggle team sharing
`/api/admin/orchestration/user-templates/:id/duplicate`	POST	Duplicate template

Method Management Endpoints:

Endpoint	Method	Description
`/api/admin/orchestration/methods`	GET	List all methods (includes `isSystemMethod`)
`/api/admin/orchestration/methods/:code`	GET	Get method details
`/api/admin/orchestration/methods/:code`	PATCH	Update method (parameters only for system methods)
`/api/admin/orchestration/metrics`	GET	Method execution metrics
`/api/admin/orchestration/executions`	GET	Recent executions

34.4 Cato Neural Decision Engine

The `CATO_NEURAL` method integrates with the Genesis Cato safety architecture:

Parameters

Parameter	Type	Default	Description
`safety_mode`	enum	`enforce`	CBF enforcement: enforce, warn, monitor
`use_affect_mapping`	boolean	`true`	Map affect state to hyperparameters
`use_predictive_coding`	boolean	`true`	Enable active inference
`precision_governor_enabled`	boolean	`true`	Limit confidence by epistemic state
`cbf_threshold`	number	`0.95`	Safety barrier threshold

Affect-to-Hyperparameter Mapping

Affect State	Hyperparameter Effect
High frustration (>0.5)	Lower temperature (more focused)
High curiosity (>0.6)	Higher temperature (exploration)
Low self-efficacy (<0.4)	Escalate to expert model
High arousal (>0.7)	Longer max tokens

34.5 Method Parameters Reference

All methods have configurable parameters that can be set at the **Admin level** (defaults) or overridden in **User Workflow Templates**.

Uncertainty & Confidence Methods

Method	Parameter	Type	Default	Description
SEMANTIC_ENTROPY	`sample_count`	integer	10	Number of response samples (5-20)
	`temperature`	number	0.7	Sampling temperature

Method	Parameter	Type	Default	Description
	`clustering_method`	enum	"nli"	Clustering: nli, embedding, exact
	`entropy_threshold`	number	0.5	Flag uncertainty above this
SE_PROBES	`probe_layers`	array	[-1, -2]	Model layers to probe (logprob-based)
	`threshold`	number	0.5	Uncertainty threshold
	`fast_mode`	boolean	true	Use fast logprob estimation
	`sample_count`	integer	5	Number of samples for averaging
KERNEL_ENTROPY	`kernel`	enum	"rbf"	Kernel: rbf, linear, polynomial
	`bandwidth`	string	"auto"	Bandwidth or "auto" for Silverman
	`sample_count`	integer	10	Response samples for KDE
CALIBRATED_CONF	`calibration_method`	enum	"platt_scaling"	platt_scaling, isotonic, temperature_scaling
	`confidence_prompt`	string	"verbalized"	How to elicit confidence
	`temperature`	number	0.3	Sampling temperature
CONSISTENCY_LIN	`sample_count`	integer	5	Number of response samples
	`agreement_metric`	enum	"jaccard"	jaccard, cosine, exact_match, bertscore
	`threshold`	number	0.7	Agreement threshold
CONFORMAL_PRED	`coverage_target`	number	0.9	Target coverage (0.5-0.99)
	`calibration_size`	integer	500	Calibration set size
	`adaptive`	boolean	true	Use adaptive conformal sets

Routing Methods

Method	Parameter	Type	Default	Description
ROUTELLM	`router_model`	enum	"matrix_factorization"	Router: matrix_factorization, bert, causal_lm
	`cost_threshold`	number	0.7	Max cost relative to baseline
	`quality_floor`	number	0.8	Minimum acceptable quality
FRUGAL_CASCADE	`model_cascade`	array	["gpt-4o-mini", "gpt-4o", "o1"]	Models in escalation order

Method	Parameter	Type	Default	Description
	`confidence_threshold`	number	0.85	Escalate below this confidence
	`max_escalations`	integer	2	Maximum escalation steps
PARETO_ROUTER	`budget_cents`	number	10	Budget constraint per query
	`quality_weight`	number	0.7	Weight for quality (0-1)
	`latency_weight`	number	0.1	Weight for latency (0-1)
C3PO_CASCADE	`cascade_levels`	integer	3	Number of model tiers
	`self_supervised`	boolean	true	Enable self-supervised learning
	`calibration_samples`	integer	100	Samples for difficulty calibration
AUTOMIX	`pomdp_horizon`	integer	3	POMDP planning horizon
	`exploration_rate`	number	0.1	for -greedy exploration
	`self_verification`	boolean	true	Verify own outputs

Debate & Deliberation Methods

Method	Parameter	Type	Default	Description
SPARSE_DEBATE	`topology`	enum	“ring”	Network: ring, star, tree, full
	`debate_rounds`	integer	3	Number of debate rounds (1-10)
	`temperature`	number	0.7	Agent response temperature
ARG_MAPPING	`strength_threshold`	number	0.5	Min argument strength to include
	`include_rebuttal`	boolean	true	Generate rebuttals
	`max_depth`	integer	3	Max argument tree depth
HAH_DELPHI	`tiers`	integer	4	Number of consensus tiers
	`human_threshold`	number	0.6	Escalate to human above this
	`consensus_target`	number	0.9	Target consensus level
	`max_rounds`	integer	5	Maximum Delphi rounds
RECONCILE_WEIGHTED	`min_confidence`	number	0.6	Minimum confidence to include
	`weight_by`	string	“confidence”	Weighting strategy
	`reconciliation_rounds`	integer	2	Reconciliation iterations

Evaluation Methods

Method	Parameter	Type	Default	Description
POLL_JUDGE	`num_judges`	integer	3	Number of judge models
	`scoring_criteria`	array	[“accuracy”, “completeness”, “clarity”]	Evaluation dimensions
	`aggregation`	enum	“mean”	Aggregation: mean, median, weighted
G_EVAL	`dimensions`	array	[“coherence”, “consistency”, “fluency”, “relevance”]	G-Eval dimensions
	`use_cot`	boolean	true	Chain-of-thought scoring
	`score_range`	array	[1, 5]	Score min/max
PAIRWISE_PREFER	`comparison_criteria`	array	[“quality”, “accuracy”, “helpfulness”]	Comparison dimensions
	`allow_tie`	boolean	true	Allow tie verdicts
SELFCHECK_GPT	`sample_count`	integer	5	Consistency check samples
	`check_type`	enum	“consistency”	Check type: consistency, bertscore, nli
	`threshold`	number	0.7	Inconsistency threshold

Hallucination Detection Methods

Method	Parameter	Type	Default	Description
MULTI_HALLUC	`methods`	array	[“consistency”, “attribution”, “semantic_entropy”]	Detection methods
	`aggregation`	enum	“weighted”	weighted, majority, any
	`flag_threshold`	number	0.6	Flag as hallucination above
METAQA	`transformations`	array	[“paraphrase”, “negation”, “entity_swap”]	Mutation types
	`num_mutations`	integer	3	Mutations per claim
	`consistency_threshold`	number	0.8	Consistency threshold
FACTUAL_GROUND	`retrieval_top_k`	integer	5	Documents to retrieve
	`evidence_threshold`	number	0.7	Evidence support threshold
	`require_explicit_support`	boolean	true	Require explicit evidence

Human-in-the-Loop Methods

Method	Parameter	Type	Default	Description
HITL_REVIEW	`confidence_threshold`	number	0.7	Route to human below this
	`stake_level`	enum	"medium"	low, medium, high, critical
	`auto_approve_above`	number	0.95	Auto-approve above this confidence
	`queue_priority`	enum	"fifo"	Queue ordering
TIERED_EVAL	`tiers`	integer	3	Evaluation tiers
	`auto_tier_threshold`	number	0.85	Auto-approve threshold
	`escalation_criteria`	array	["low_confidence", "high_stakes"]	When to escalate
ACTIVE_SAMPLE	`uncertainty_method`	enum	"entropy"	entropy, margin, random
	`batch_size`	integer	10	Samples per batch
	`diversity_weight`	number	0.3	Diversity in selection

34.6 User Workflow Template Parameter Overrides

When users create workflow templates, they can override default parameters for each step:

```
// Example: User template with parameter overrides
{
  "templateName": "High-Confidence Research",
  "steps": [
    {
      "stepOrder": 1,
      "methodCode": "SEMANTIC_ENTROPY",
      "parameters": {
        "sample_count": 15,        // Override: more samples
        "entropy_threshold": 0.3   // Override: stricter threshold
      }
    },
    {
      "stepOrder": 2,
      "methodCode": "FRUGAL_CASCADE",
      "parameters": {
        "confidence_threshold": 0.95,  // Override: higher confidence
        "max_escalations": 3           // Override: allow more escalation
      }
    }
  ]
}
```

Admin Dashboard (`Orchestration → Methods`): - View and edit **default parameters** for all methods - Changes apply to all workflows using the method - System methods: parameters only,

not method definition

Think Tank UI (`Workflow Templates`): - Users create templates with **parameter overrides** - Overrides apply only to that template - Templates can be shared with team

34.7 Database Tables

```sql
-- Methods with display/scientific names
ALTER TABLE orchestration_methods
ADD COLUMN display_name VARCHAR(200),
ADD COLUMN scientific_name VARCHAR(300),
ADD COLUMN research_reference TEXT,
ADD COLUMN accuracy_improvement VARCHAR(200),
ADD COLUMN complexity_level VARCHAR(50);

-- User workflow templates
CREATE TABLE user_workflow_templates (
  template_id UUID PRIMARY KEY,
  tenant_id UUID NOT NULL,
  user_id UUID NOT NULL,
  template_name VARCHAR(200) NOT NULL,
  template_description TEXT,
  base_workflow_code VARCHAR(100),
  steps JSONB NOT NULL DEFAULT '[]',
  category VARCHAR(100),
  tags TEXT[] DEFAULT '{}',
  is_shared BOOLEAN DEFAULT false,
  times_used INTEGER DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(tenant_id, user_id, template_name)
);
```

34.7 Implementation Files

File	Purpose
`migrations/066_orchestration_patterns_registry.sql`	Base registry with `is_system_method/is_system_workflow`
`migrations/157_orchestration_methods_part1.sql`	Schema updates, display/scientific names
`migrations/157_orchestration_methods_part2.sql`	Essential verification methods
`migrations/157_orchestration_methods_part3.sql`	Uncertainty routing, neural methods
`lambda/shared/services/orchestration/methods_impl.py`	**20 algorithm implementations** including SE Probes, Kernel Entropy, Pareto, C3PO, AutoMix
`lambda/shared/services/cato/neural_decision_engine.py`	Cato Neural Decision Engine
`lambda/admin/orchestration-methods/index.py`	Methods API with system method protection
`lambda/admin/orchestration-user-templates/index.py`	User templates CRUD API
`apps/admin-dashboard/app/(dashboard)/orchestration/methods/page.tsx`	Admin configuration page
`apps/admin-dashboard/app/(dashboard)/thinktank/workflow-templates/page.tsx`	Think Tank UI

35. Polymorphic UI (PROMPT-41)

35.1 Overview

Flowise outputs Text. Think Tank outputs Applications.

The Polymorphic UI system makes Think Tank's interface physically transform based on task complexity, domain hints, and drive profile. Unlike static chatbot interfaces, Think Tank morphs into the tool the user actually needs.

35.2 The Gearbox (Elastic Compute)

Users can manually control the cost-quality tradeoff via the Gearbox:

Mode	Cost	Architecture	Memory	Use Case
Sniper	$0.01/run	Single Model	Read-Only Ghost Memory	Quick answers, lookups, coding
War Room	$0.50+/run	Multi-Agent Ensemble	Read/Write + Active Inference	Strategy, audits, reasoning

Escalation: A green "Escalate to War Room" button appears after Sniper responses.

35.3 The Three Views

View	Intent	Morph	Key Feature
Sniper	Quick commands	Terminal/Command Center	Green badge, cost transparency, immediate execution
Scout	Research & exploration	Infinite Canvas/Mind Map	Sticky notes, topic clustering, conflict lines
Sage	Audit & validation	Split-Screen Diff Editor	Left=content, Right=sources with confidence scores

35.4 View Types

View Type	Trigger	Description
`terminal_simple`	Quick commands, lookups	Command Center - fast execution
`mindmap`	Research, exploration	Infinite Canvas - visual mapping
`diff_editor`	Verification, compliance	Split-Screen - source validation
`dashboard`	Analytics queries	Metrics visualization
`decision_cards`	HITL escalation	Mission Control interface
`chat`	Default	Standard conversation

35.5 Configuration

Access via **Think Tank → Polymorphic UI** in admin dashboard.

Setting	Description	Default
`enableAutoMorphing`	Auto-morph based on query	`true`
`enableGearboxToggle`	Show Sniper/War Room toggle	`true`
`enableCostDisplay`	Show cost badges	`true`
`enableEscalationButton`	Show Escalate button	`true`
`defaultExecutionMode`	Default mode	`sniper`
`domainViewOverrides`	Per-domain view mapping	medical/financial/legal → `diff_editor`

35.6 Implementation Files

File	Purpose
`governor/economic-governor.ts`	`determineViewType()`, `determinePolymorphicRoute()`
`consciousness/mcp-server.ts`	`render_interface`, `escalate_to_war_room` tools
`python/cato/cognitive/workflows.py`	Flyte tasks for view selection
`migrations/160_polymorphic_ui.sql`	Database schema
`components/thinktank/polymorphic/`	React view components
`app/(dashboard)/thinktank/polymorphic/page.tsx`	Admin page

35.7 Database Tables

Table	Purpose
`view_state_history`	Tracks UI morphing decisions
`execution_escalations`	Tracks Sniper → War Room escalations
`polymorphic_config`	Per-tenant configuration

35.8 API Endpoints

Base: `/api/admin/polymorphic`

Method	Endpoint	Description
GET	`/?action=config`	Get configuration
GET	`/?action=view-history`	Get view state history
GET	`/?action=escalations`	Get escalation history
GET	`/?action=analytics`	Get usage analytics
POST	`/` (action: render)	Render specific view
POST	`/` (action: escalate)	Escalate to War Room
POST	`/` (action: update-config)	Update configuration

36. Think Tank Policy Framework: Strategic Intelligence

The Think Tank is not merely a chatbot interface—it is a **policy research engine** informed by rigorous analysis from organizations like the Cato Institute. This section documents the strategic framework that underlies Think Tank's approach to technology policy, regulation, and economic analysis.

36.1 The Cato Institute Policy Foundation

The Cato Institute, a renowned policy think tank, provides crucial insights on technology regulation and innovation. Think Tank integrates these perspectives to ensure users receive balanced, evidence-based analysis rather than ideologically-driven conclusions.

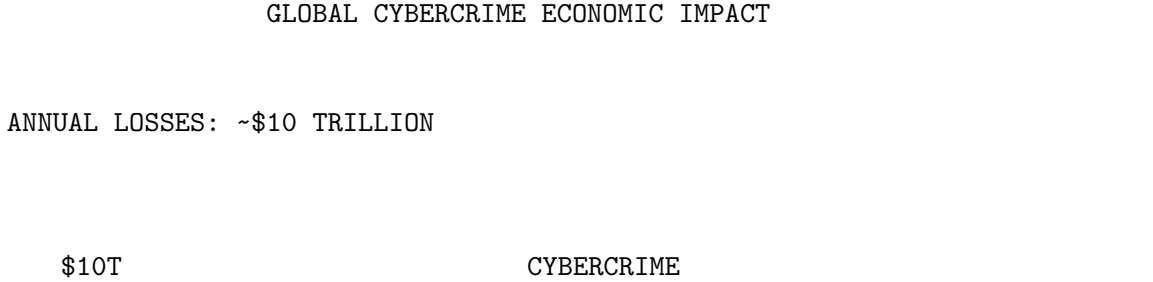
Core Policy Principles

Principle	Description	Application in Think Tank
Pro-Innovation Default	Permissionless innovation should be the norm	Bias toward "how can this work?" not "why won't this work?"
Evidence-Based Analysis	Policy must be grounded in data, not fear	Require citations and data sources in policy discussions
Regulatory Humility	Acknowledge limits of prediction	Flag uncertainty explicitly; avoid overconfident prescriptions
Market-Based Solutions	Private sector often innovates faster than regulators	Explore voluntary standards before mandates
Individual Liberty	Technology should enhance freedom, not surveillance	Privacy-by-design; minimize data collection

36.2 The $10 Trillion Cybercrime Economy

A critical context for Think Tank's policy analysis is the scale of the threat environment. The global economy loses approximately **$10 trillion annually** to cybercrime—a figure larger than the GDP of every country except the United States and China.

Economic Impact Analysis



```
$4.2T               GERMANY GDP

$3.4T             JAPAN GDP

$2.1T          UK GDP
```

```
BREAKDOWN BY ATTACK TYPE:
```

- `Ransomware: $20B+ annually`
- `Business Email Compromise: $2.7B annually`
- `Data Breaches: $4.45M average cost per incident`
- `Supply Chain Attacks: Growing 742% since 2019`
- `Nation-State Attacks: Incalculable strategic damage`

Implications for Policy Discussion When users ask Think Tank about cybersecurity policy, the system contextualizes recommendations against this $10T backdrop:

User Query	Policy-Aware Response Approach
"Should we regulate AI?"	Frame against: AI can *reduce* $10T losses if deployed correctly
"Is this security spending justified?"	Compare against: Your $1M security budget vs. $4.45M average breach cost
"Should we ban ransomware payments?"	Analyze: Prohibition vs. harm reduction strategies
"How strict should compliance be?"	Balance: Compliance costs vs. breach probability × impact

36.3 Memory Safety and the 70% Problem

The Think Tank injects a specific technical insight into relevant conversations: **70% of all software vulnerabilities** stem from memory safety issues. This statistic, validated by Microsoft, Google, and the NSA, should inform every software architecture discussion.

Memory Safety Vulnerability Classes

Vulnerability	Description	% of CVEs
Buffer Overflow	Writing beyond allocated memory	~30%
Use-After-Free	Accessing freed memory	~20%
Double Free	Freeing memory twice	~8%
Null Pointer Dereference	Accessing null pointers	~7%
Integer Overflow	Arithmetic exceeding bounds	~5%
Total Memory Safety	All memory-related issues	**~70%**

Policy Recommendation Engine When users discuss software architecture, security, or procurement, Think Tank can inject policy-informed guidance:

```
// Policy injection for memory safety discussions
const MEMORY_SAFETY_POLICY = {
  context: 'User discussing software architecture or security',

  injectedGuidance: `
    POLICY CONTEXT: Memory safety vulnerabilities account for 70% of all CVEs.

    RECOMMENDATIONS:
    1. For new projects: Prefer memory-safe languages (Rust, Go, Swift)
    2. For existing C/C++ code: Consider incremental migration or sandboxing
    3. For procurement: Require memory-safe language attestation from vendors
    4. For risk assessment: Memory-unsafe components = higher risk weight

    SOURCE: Microsoft, Google, NSA research (2019-2024)
  `,

  triggerPatterns: [
    'architecture', 'security', 'language choice', 'procurement',
    'vulnerability', 'CVE', 'buffer overflow', 'memory'
  ],
};
```

36.4 Regulatory Stance Configuration

Administrators can configure Think Tank's default policy stance for their organization:

Setting	Options	Description
`defaultRegulatorystance`	`cautious` / `balanced` / `pro_innovation`	Bias in regulatory discussions
`requireCitations`	`true` / `false`	Require sources for policy claims
`flagUncertainty`	`always` / `when_high` / `never`	Uncertainty disclosure level
`privateDataBias`	`minimize` / `balanced` / `maximize_utility`	Data collection philosophy
`complianceEmphasis`	`strict` / `risk_based` / `minimal`	Compliance recommendation style

Configuration Example Navigate to **Think Tank → Policy Framework** in admin dashboard:

```
# policy-framework.config.yaml
policy_framework:
  enabled: true

  default_stance: balanced
```

```yaml
cybercrime_context:
  enabled: true
  inject_10T_context: true
  inject_memory_safety_context: true

citation_requirements:
  require_for_policy_claims: true
  preferred_sources:
    - cato_institute
    - brookings
    - nist
    - academic_peer_reviewed
  flag_opinion_vs_fact: true

uncertainty_handling:
  disclosure_level: always
  confidence_threshold_for_recommendation: 0.7
  escalate_low_confidence_to_human: true

privacy_settings:
  default_data_collection: minimize
  require_purpose_limitation: true
  support_right_to_deletion: true
```

36.5 Database Tables

Table	Purpose
`policy_framework_config`	Per-tenant policy configuration
`policy_citation_log`	Citations used in policy discussions
`policy_uncertainty_flags`	Uncertainty disclosures
`regulatory_stance_overrides`	Per-domain stance overrides

36.6 Implementation Files

File	Purpose
`lambda/shared/services/policy-framework.ts`	Policy context injection
`lambda/shared/services/citation-manager.ts`	Citation tracking and validation
`lambda/thinktank/policy-context.ts`	API handler for policy queries
`migrations/164_policy_framework.sql`	Database schema
`config/policy/cato-principles.yaml`	Cato Institute policy principles

37. Agentic Orchestration: SSF, CAEP, and Identity Remediation

Think Tank's AI agents operate within a rigorous security framework that leverages open standards for real-time security signaling. This section documents the Shared Signals Framework (SSF) and Continuous Access Evaluation Profile (CAEP) integration specific to Think Tank operations.

37.1 The Agentic AI Paradigm

Traditional automation follows rigid if-then rules. **Agentic AI** represents a fundamental shift: AI systems that continuously evaluate their environment, learn from outcomes, and adapt their behavior within defined safety constraints.

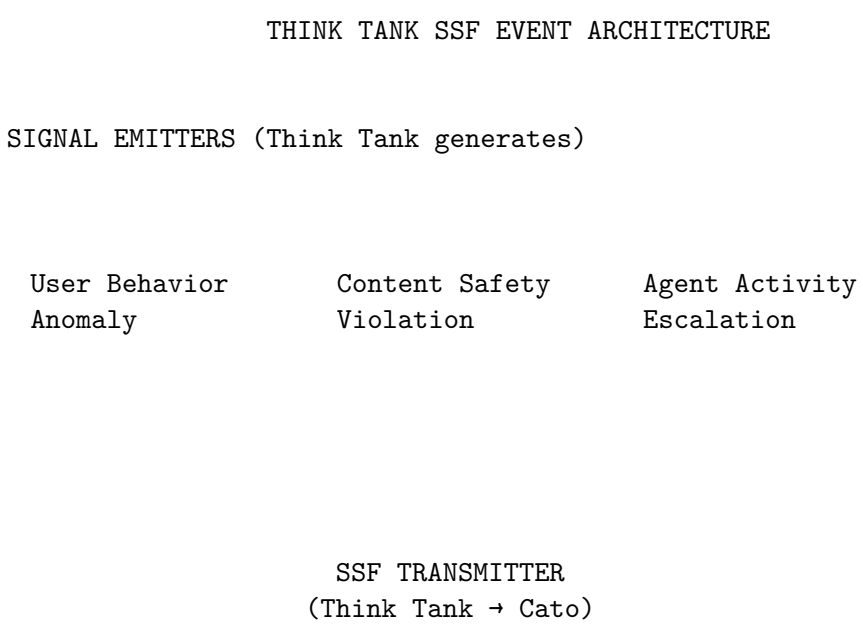
Traditional Automation vs. Agentic AI

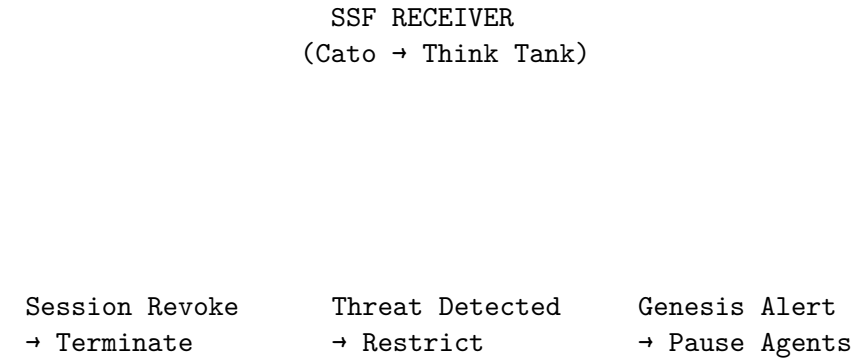
Aspect	Traditional Automation	Agentic AI (Think Tank)
Decision Logic	Static rules	Dynamic evaluation
Learning	None	Continuous adaptation
Error Handling	Fail or retry	Investigate and adapt
Human Interaction	Scheduled checkpoints	On-demand escalation
Security Model	Perimeter-based	Zero Trust + CAEP

37.2 Shared Signals Framework (SSF) Integration

The **Shared Signals Framework** is an open standard that enables real-time security event sharing between systems. Think Tank agents both emit and consume SSF signals.

SSF Event Flow in Think Tank





SSF Events Emitted by Think Tank

Event Type	Trigger	Recipient Action
`thinktank.behavior.anomaly`	User behavior deviates significantly from baseline	Cato increases monitoring
`thinktank.content.violation`	User attempts prohibited content	Identity provider notified
`thinktank.agent.escalation`	Agent requires human approval	Mission Control alerted
`thinktank.session.suspicious`	Multiple failed attempts or unusual patterns	Cato may revoke session
`thinktank.data.exfiltration_attempt`	Suspected data extraction attempt	Block and investigate

SSF Events Consumed by Think Tank

Event Type	Source	Think Tank Action
`session-revoked`	Identity Provider	Immediately terminate user session
`credential-change`	Identity Provider	Force re-authentication
`threat-detected`	Cato SASE	Restrict agent capabilities
`genesis.alert`	Genesis Reactor	Pause all non-critical agents
`device-compliance-change`	MDM/EDR	Re-evaluate user permissions

37.3 Continuous Access Evaluation Profile (CAEP)

CAEP extends SSF with specific event types designed for continuous session validation. Unlike traditional session timeouts, CAEP enables **real-time session adjustment** based on security signals.

CAEP Session Lifecycle

```
                    CAEP SESSION LIFECYCLE IN THINK TANK


   1. SESSION START

       User authenticates → Session created → CAEP listener registered


   2. CONTINUOUS EVALUATION (every CAEP event)


         CAEP Event Received → Evaluate Impact → Adjust Session

         Example: IP Change
           Same country? → Log only
           Different country? → Step-up authentication
           Impossible travel? → Terminate session



   3. SESSION TERMINATION

       CAEP revoke signal OR user logout OR timeout → Clean termination
       → Preserve conversation state → Clear credentials → Audit log
```

CAEP Configuration for Think Tank

```yaml
# caep-thinktank.config.yaml
caep:
  enabled: true

  session_management:
    # How to handle IP changes
    ip_change_policy:
      same_country: log_only
      different_country: step_up_auth
      impossible_travel: terminate_session
```

```
      impossible_travel_threshold_hours: 2

    # How to handle device changes
    device_change_policy:
      known_device: allow
      unknown_device_trusted_location: step_up_auth
      unknown_device_unknown_location: terminate_session

    # How to handle credential events
    credential_change_policy:
      password_change: force_reauth
      mfa_change: force_reauth
      role_change: re_evaluate_permissions

  agent_restrictions:
    # During security events, restrict agent capabilities
    during_threat_detected:
      disable_autonomous_actions: true
      disable_external_api_calls: true
      require_human_approval_all: true

    during_genesis_alert:
      pause_all_agents: true
      preserve_state: true
      notify_administrators: true
```

37.4 Autonomous Identity Remediation

Think Tank agents can perform **autonomous identity remediation**—cleaning up identity data quality issues without human intervention. This capability requires careful configuration to balance efficiency with safety.

Remediation Capabilities

Action	Autonomous?	Conditions
Remove Orphan Account	Yes	Inactive 90+ days, no entitlements, no recent auth
Disable Stale Service Account	Yes	Unused 180+ days, not in critical systems
Fix Group Membership Inconsistency	Yes	Source of truth mismatch detected
Delete Human Account	**No**	Always requires human approval
Revoke Admin Privileges	**No**	Always requires human approval

Action	Autonomous?	Conditions
Bulk Operations (100+)	**No**	Always requires human approval

Remediation Workflow

```
                    AUTONOMOUS REMEDIATION WORKFLOW


   1. DETECTION
      Agent scans Identity Fabric for anomalies
        Orphan accounts (no manager, no recent activity)
        Stale service accounts (unused beyond threshold)
        Inconsistent group memberships (source mismatch)

   2. VALIDATION
        Verify anomaly against multiple data sources
        Check against exclusion lists (VIPs, system accounts)
        Confirm remediation action is within autonomous scope
        Verify Genesis Interlock allows action (no active alerts)

   3. EXECUTION
        If autonomous: Execute immediately, log action
        If requires approval: Create Mission Control escalation

   4. VERIFICATION
        Confirm action completed successfully
        Verify no unintended side effects
        Update remediation statistics

   5. LEARNING
        If action succeeded: Reinforce pattern
        If action failed or reversed: Flag for human review
```

Remediation Configuration Navigate to **Think Tank → Identity → Remediation** in admin dashboard:

```yaml
# identity-remediation-thinktank.config.yaml
identity_remediation:
  enabled: true

  # Autonomous action limits
  autonomous_limits:
    max_per_hour: 50
```

```yaml
  max_per_day: 200
  cooldown_after_error_minutes: 30

# Detection criteria
detection:
  orphan_account:
    inactive_days_threshold: 90
    require_no_entitlements: true
    require_no_recent_auth: true
    exclude_patterns:
      - "*-system@*"
      - "*-service@*"
      - "admin@*"

  stale_service_account:
    unused_days_threshold: 180
    exclude_critical_systems: true
    critical_system_tags:
      - genesis
      - security
      - auth

  group_membership:
    check_source_of_truth: true
    sources:
      - active_directory
      - scim_provider
      - hr_system

# Notification settings
notifications:
  notify_on_autonomous_action: true
  notify_recipients:
    - security_team
    - identity_admins
  daily_summary: true

# Genesis Interlock integration
genesis_interlock:
  pause_during: [GEN-300, GEN-400, GEN-500]
  resume_automatically: true
```

37.5 The Radiant Ghost in Think Tank

The **Radiant Ghost** metaphor extends to Think Tank's user interface, providing visual feedback about agent activity.

Ghost States in Think Tank UI

State	Indicator	Meaning	User Action
Idle	Faint outline	No active agents	None
Thinking	Soft pulse	Agent processing request	Wait
Researching	Searching animation	Agent gathering information	Wait
Writing	Typing animation	Agent generating response	Watch
Validating	Checkmark animation	Agent verifying output	Wait
Escalating	Orange pulse	Agent needs human input	Respond
Alert	Red pulse	Security or safety event	Investigate

Ghost Configuration

```yaml
# ghost-ui.config.yaml
ghost_ui:
  enabled: true

  # Visual settings
  visuals:
    idle_opacity: 0.3
    active_opacity: 0.8
    alert_opacity: 1.0
    animation_speed: normal  # slow, normal, fast

  # State display
  states:
    show_thinking: true
    show_researching: true
    show_writing: true
    show_validating: true
    show_escalating: true
    show_alert: true

  # Accessibility
  accessibility:
    respect_reduced_motion: true
    provide_text_status: true
    screen_reader_announcements: true
```

37.6 Database Tables

Table	Purpose
`ssf_events_emitted`	SSF events sent by Think Tank
`ssf_events_received`	SSF events received by Think Tank
`caep_session_events`	CAEP session lifecycle events
`identity_remediation_log`	All remediation actions
`identity_remediation_errors`	Failed remediation attempts
`ghost_state_log`	Ghost UI state transitions

37.7 API Endpoints

Base: `/api/thinktank/security`

Method	Endpoint	Description
GET	`/ssf/status`	SSF integration status
GET	`/ssf/events`	Recent SSF events
POST	`/ssf/emit`	Manually emit SSF event (admin only)
GET	`/caep/session`	Current session CAEP status
GET	`/remediation/stats`	Remediation statistics
GET	`/remediation/log`	Remediation action log
POST	`/remediation/trigger`	Trigger manual remediation scan
GET	`/ghost/state`	Current Ghost state

37.8 Implementation Files

File	Purpose
`lambda/shared/services/ssf-thinktank-service.ts`	SSF transmitter/receiver for Think Tank
`lambda/shared/services/caep-session-service.ts`	CAEP session management
`lambda/shared/services/identity-remediation-service.ts`	Think Tank-specific remediation logic
`lambda/thinktank/security-events.ts`	Security event API handler
`components/thinktank/ghost-indicator.tsx`	Ghost UI component
`migrations/165_agentic_orchestration.sql`	Database schema
`config/security/ssf-events.yaml`	SSF event definitions
`config/security/caep-policies.yaml`	CAEP policy configuration

---

38. Advanced Features (v4.18.0)

This section covers new advanced features implemented for Think Tank.

38.1 Flash Facts (Knowledge Sparks)

Fast-access factual memory system for quick retrieval of verified facts.

UI Metaphor: "Knowledge Sparks" - Contextual sidebar widget showing relevant facts as glowing spark icons.

Features: - CRUD operations for facts with confidence scoring - Semantic search using vector embeddings - Automatic fact extraction from conversations - Fact verification workflow - Usage tracking and statistics

API Endpoints (Base: `/api/thinktank/flash-facts`):

Method	Endpoint	Description
GET	`/`	List facts with filtering
POST	`/`	Create a new fact
GET	`/:id`	Get a specific fact
PUT	`/:id`	Update a fact
DELETE	`/:id`	Delete a fact
POST	`/query`	Semantic search for facts
POST	`/extract`	Extract facts from conversation
POST	`/:id/verify`	Verify a fact
GET	`/stats`	Get usage statistics

Database Tables: - `flash_facts` - Fact storage with embeddings - `flash_facts_config` - Per-tenant configuration

---

38.2 Grimoire (Spell Book)

Procedural memory system for storing and executing reusable patterns.

UI Metaphor: "Spell Book" - Magical tome with spell cards organized by schools of magic.

Schools of Magic: | School | Icon | Purpose | |——–|——|———| | Code | | Programming patterns | | Data | | Data manipulation | | Text | | Text processing | | Analysis | | Analytical methods | | Design | | UI/UX patterns | | Integration | | API integrations | | Automation | | Workflow automation | | Universal | | Cross-domain spells |

Spell Categories: - Transformation, Divination, Conjuration, Abjuration - Enchantment, Illusion, Necromancy, Evocation

API Endpoints (Base: `/api/thinktank/grimoire`):

Method	Endpoint	Description
GET	`/`	Get grimoire overview
GET	`/spells`	List all spells
POST	`/spells`	Create a new spell
GET	`/spells/:id`	Get spell details
PUT	`/spells/:id`	Update a spell
DELETE	`/spells/:id`	Delete a spell
POST	`/spells/:id/cast`	Cast a spell
POST	`/spells/:id/learn`	Learn from failure
GET	`/schools`	Get schools of magic
GET	`/categories`	Get spell categories

Method	Endpoint	Description
POST	`/match`	Find matching spell
POST	`/promote`	Promote pattern to spell

38.3 Economic Governor (Fuel Gauge)

Model arbitrage and cost optimization system.

UI Metaphor: "Fuel Gauge" - Visual meter showing budget remaining with color-coded status.

Governor Modes: | Mode | Description | |——|————-| | `cost_minimizer` | Always cheapest viable option | | `quality_maximizer` | Best quality within budget | | `balanced` | Balance cost and quality | | `latency_focused` | Prioritize response speed | | `custom` | Use custom arbitrage rules |

Model Tiers: - Economy ( ) - Cheapest, simple tasks - Self-Hosted ( ) - On-premise models - Standard ( ) - Default tier - Premium ( ) - Higher quality - Flagship ( ) - Best available

API Endpoints (Base: `/api/thinktank/governor`):

Method	Endpoint	Description
GET	`/`	Get dashboard with fuel gauge
GET	`/config`	Get configuration
PUT	`/config`	Update configuration
PUT	`/mode`	Quick mode switch
POST	`/recommend`	Get model recommendation
GET	`/metrics`	Get cost metrics
GET	`/budget`	Get budget status
PUT	`/budget`	Update budget limit
GET	`/tiers`	Get model tiers
PUT	`/tiers/:tier`	Update tier config
GET	`/rules`	Get arbitrage rules
POST	`/rules`	Add arbitrage rule
PUT	`/rules/:id`	Update rule
DELETE	`/rules/:id`	Delete rule

38.4 Sentinel Agents (Watchtower Dashboard)

Event-driven autonomous agents for monitoring and automation.

UI Metaphor: "Watchtower Dashboard" - Castle towers watching over different domains.

Agent Types: | Type | Icon | Purpose | |——|——|———| | Monitor | | Passive watchdog - observes and alerts | | Guardian | | Protective - can block harmful actions | | Scout | | Proactive information gathering | | Herald | | Notifications and announcements | | Arbiter | | Decision making and routing |

Agent Status: - Idle ( ), Watching ( ), Triggered ( ), Cooldown ( ), Disabled ( )

API Endpoints (Base: `/api/thinktank/sentinels`):

Method	Endpoint	Description
GET	`/`	List all agents
POST	`/`	Create agent
GET	`/:id`	Get agent details
PUT	`/:id`	Update agent
DELETE	`/:id`	Delete agent
POST	`/:id/trigger`	Manually trigger
POST	`/:id/enable`	Enable agent
POST	`/:id/disable`	Disable agent
GET	`/:id/events`	Get agent events
GET	`/events`	All events
GET	`/stats`	Statistics
GET	`/types`	Available types

38.5 Time-Travel Debugging (Timeline Scrubber)

Conversation forking and state replay system.

UI Metaphor: "Timeline Scrubber" - Horizontal timeline with draggable playhead and fork points.

Checkpoint Types: | Type | Icon | Description | |——|——|————-| | Auto | | Automatic checkpoint | | Manual | | User-created | | Fork | | Branch point | | Merge | | Merged timelines | | Rollback | | After rollback |

API Endpoints (Base: `/api/thinktank/time-travel`):

Method	Endpoint	Description
GET	`/timelines`	List timelines
POST	`/timelines`	Create timeline
GET	`/timelines/:id`	Get timeline view
POST	`/timelines/:id/checkpoint`	Create checkpoint
POST	`/timelines/:id/jump`	Jump to checkpoint
POST	`/timelines/:id/fork`	Fork timeline
POST	`/timelines/:id/replay`	Replay sequence
GET	`/checkpoints/:id`	Get checkpoint

38.6 Council of Rivals (Debate Arena)

Multi-model adversarial consensus system.

UI Metaphor: "Debate Arena" - Amphitheater with model avatars debating in a circular arrangement.

Member Roles: | Role | Icon | Purpose | |——|——|———| | Advocate | | Argues for a position | | Critic | | Challenges positions | | Synthesizer | | Combines perspectives | | Specialist | | Domain expert | | Contrarian | | Devil's advocate |

Preset Councils: - **Balanced** ( ) - Diverse perspectives - **Technical** ( ) - Expert technical review - **Creative** ( ) - Creative exploration

Verdict Outcomes: - Consensus ( ), Majority ( ), Split ( ), Deadlock ( ), Synthesized ( )

API Endpoints (Base: `/api/thinktank/council`):

Method	Endpoint	Description
GET	`/`	List councils
POST	`/`	Create council
POST	`/preset`	Create preset council
GET	`/:id`	Get council
PUT	`/:id`	Update council
DELETE	`/:id`	Delete council
POST	`/:id/debate`	Start debate
GET	`/debate/:id`	Get debate
POST	`/debate/:id/argument`	Submit argument
POST	`/debate/:id/rebuttal`	Submit rebuttal
POST	`/debate/:id/vote`	Conduct voting
GET	`/presets`	Get preset options

38.7 Security Signals (Security Shield)

SSF/CAEP integration for identity security events.

UI Metaphor: "Security Shield" - Animated shield with real-time threat visualization.

Signal Types: | Type | Icon | Description | |——|——|————-| | Session Revoked | | SSF session terminated | | Credential Change | | Password/key changed | | Device Compliance | | CAEP compliance change | | Risk Change | | Risk level changed | | Anomaly Detected | | Behavioral anomaly | | Threat Detected | | Active threat | | Policy Violation | | Policy breach |

Severity Levels: Critical ( ), High ( ), Medium ( ), Low ( ), Info ( )

API Endpoints (Base: `/api/thinktank/security`):

Method	Endpoint	Description
GET	`/dashboard`	Security dashboard
GET	`/signals`	List signals
POST	`/signals`	Create signal
GET	`/signals/:id`	Get signal
PUT	`/signals/:id/status`	Update status

Method	Endpoint	Description
GET	`/policies`	List policies
POST	`/policies`	Create policy
PUT	`/policies/:id`	Update policy
DELETE	`/policies/:id`	Delete policy
POST	`/ssf/event`	Ingest SSF event
POST	`/caep/event`	Ingest CAEP event

38.8 Policy Framework (Stance Compass)

Strategic intelligence and regulatory stance configuration.

UI Metaphor: "Stance Compass" - Radial chart showing policy positions across domains.

Policy Domains: | Domain | Icon | Description | |———|——|————-| | AI Safety | | AI alignment and safety | | Data Privacy | | Data protection regulations | | Content Moderation | | Content policies | | Accessibility | | Accessibility requirements | | Sustainability | | Environmental considerations | | Security | | Cybersecurity posture | | Transparency | | AI transparency | | Ethics | | Ethical AI principles | | Compliance | | Regulatory compliance | | Innovation | | Innovation balance |

Stance Positions: - Restrictive ( ), Cautious ( ), Balanced ( ), Permissive ( ), Adaptive ( )

Preset Profiles: - **Conservative** - Maximum safety, minimal risk - **Balanced** - Middle ground - **Innovative** - Emphasis on innovation

API Endpoints (Base: `/api/thinktank/policy`):

Method	Endpoint	Description
GET	`/compass`	Get compass view
GET	`/domains`	Get available domains
GET	`/positions`	Get stance positions
GET	`/stances`	List stances
POST	`/stances`	Create stance
GET	`/stances/:domain`	Get domain stance
PUT	`/stances/:id`	Update stance
GET	`/profiles`	List profiles
GET	`/profiles/active`	Get active profile
POST	`/profiles`	Create profile
POST	`/profiles/preset`	Create preset profile
PUT	`/profiles/:id/activate`	Activate profile
GET	`/recommendations`	Get recommendations
GET	`/compliance`	Check compliance

38.9 Database Migration

All features use migration `100_thinktank_advanced_features.sql` with the following tables:

Table	Feature
`flash_facts`	Flash Facts storage
`flash_facts_config`	Flash Facts config
`grimoire_spells`	Grimoire spells
`grimoire_casts`	Spell cast history
`grimoire_achievements`	User achievements
`economic_governor_config`	Governor config
`economic_governor_usage`	Usage tracking
`sentinel_agents`	Sentinel agents
`sentinel_events`	Agent events
`time_travel_timelines`	Timelines
`time_travel_checkpoints`	Checkpoints
`time_travel_forks`	Fork records
`council_of_rivals`	Councils
`council_debates`	Debates
`security_signals`	Security signals
`security_policies`	Security policies
`policy_stances`	Policy stances
`policy_profiles`	Policy profiles

---

39. Liquid Interface (Generative UI)

39.1 Overview

"Don't Build the Tool. BE the Tool."

The Liquid Interface transforms the chat interface into dynamic, morphable UI tools based on user intent. Instead of asking "help me make a spreadsheet app," the chat *becomes* the spreadsheet.

Core Concept: - User says: "Help me track my invoices" - Chat morphs into: Invoice tracker with data grid, totals, AI assistant sidebar - User interacts: Add, edit, filter invoices directly - AI watches: Ghost State binds UI actions to AI context - Export: "Eject" to a deployable Next.js/Vite app

Key Benefits: - **Zero-friction prototyping** - Ideas become tools instantly - **Two-way AI binding** - AI sees what you're doing, UI reflects what AI knows - **Production export** - Ephemeral apps become real codebases - **50+ morphable components** - Data grids, charts, kanban, calendars, code editors

---

39.2 Architecture

```
User Message
```

```
                Intent Detection

 data_analysis     tracking      visualization    ...




                Schema Generation

 LiquidSchema {
   layout: { type: 'split', children: [...] }
   bindings: [{ sourceComponent, contextKey, direction }]
   aiOverlay: { mode: 'sidebar', position: 'right' }
 }




                Liquid Renderer

DataGrid    Chart      Kanban    AI Chat




                Ghost State Manager
UI Events    AI Context
AI Updates    UI Components
```

39.3 Component Registry

50+ pre-built morphable components across 9 categories:

Category	Components	Description
Data (10)	DataGrid, PivotTable, DataCard, JSONViewer, SQLViewer, CSVEditor, DataFilter, SchemaDesigner, DataDiff, DataImport	Spreadsheets, tables, data viewers
Visualization (10)	LineChart, BarChart, PieChart, ScatterPlot, AreaChart, Heatmap, Treemap, GeoMap, Timeline, SankeyDiagram	Charts, graphs, maps
Productivity (10)	KanbanBoard, Calendar, GanttChart, TodoList, NotesEditor, Timer, HabitTracker, MindMap, Whiteboard, FileManager	Task & project management
Finance (6)	Invoice, BudgetTracker, ExpenseTable, Portfolio, Calculator, CurrencyConverter	Financial tools
Code (6)	CodeEditor, Terminal, DiffViewer, APITester, RegexTester, JSONFormatter	Developer tools
AI (4)	AIChat, InsightCard, SuggestionPanel, ContextInspector	AI-powered widgets
Input (4)	FormBuilder, SliderPanel, DateRangePicker, SearchBox	User input forms

Component Definition:

```
interface LiquidComponent {
  id: string;                    // e.g., 'data-grid'
  name: string;                  // e.g., 'DataGrid'
  category: ComponentCategory;   // e.g., 'data'
  propsSchema: JSONSchema;       // Component props
  eventsSchema: JSONSchema;      // Emitted events
  supportsInteraction: boolean;
  supportsDataBinding: boolean;
  supportsAIContext: boolean;
  defaultSize: { width, height };
  icon: string;
  tags: string[];
}
```

39.4 Ghost State (Two-Way Binding)

"The AI sees what you're doing. The UI reflects what AI knows."

Ghost State creates bidirectional bindings between UI components and AI context:

```
   UI Component                  AI Context

 selectedRow: 5      GhostBinding      user_selection
 filterValue: X                 applied_filter

 [AI updates]        AI Reaction       insight: "..."
```

Binding Configuration:

```typescript
interface GhostBinding {
  id: string;
  sourceComponent: string;    // UI component ID
  sourceProperty: string;     // e.g., 'selectedRow'
  contextKey: string;         // AI context key
  direction: 'ui_to_ai' | 'ai_to_ui' | 'bidirectional';
  debounceMs?: number;        // Debounce rapid changes
  triggerReaction?: boolean;  // Trigger AI response on change
  reactionPrompt?: string;    // Custom prompt for reaction
}
```

AI Reactions: When users interact with morphed UI, the AI can react with: - `speak` - Send a message - `update` - Update UI state - `morph` - Transform to different layout - `suggest` - Show suggestion panel

---

39.5 Intent Detection

Intent detection determines when and how to morph the UI:

Intent Category	Trigger Phrases	Default Components
`data_analysis`	spreadsheet, excel, csv, analyze data	DataGrid, PivotTable
`tracking`	track invoices, manage expenses	Invoice, ExpenseTable, Kanban
`visualization`	chart, graph, visualize, show metrics	LineChart, BarChart, Dashboard
`planning`	plan project, timeline, kanban	KanbanBoard, GanttChart
`calculation`	calculate, compute, formula	Calculator, DataGrid
`design`	design, wireframe, brainstorm	Whiteboard, MindMap
`coding`	code, debug, terminal	CodeEditor, Terminal
`writing`	write, draft, notes	NotesEditor, MindMap

Morph Threshold: `confidence >= 0.85` (configurable per tenant)

39.6 Eject to App

"The Takeout Button" - Export ephemeral liquid apps to real codebases.

Supported Frameworks: - **Next.js 14** - Full-stack React with API routes - **Vite + React** - Fast client-side SPA - **Remix** - Web standards framework - **Astro** - Content-focused sites

Features to Include: | Feature | Description | |———|————-| | `database` | PGLite → Postgres migration, Drizzle ORM | | `auth` | NextAuth scaffolding | | `api` | API routes for data operations | | `ai` | OpenAI integration | | `realtime` | WebSocket support |

Generated Files:

```
my-liquid-app/
  package.json
  next.config.mjs / vite.config.ts
  tsconfig.json
  tailwind.config.ts
  app/page.tsx (or src/App.tsx)
  components/
     LiquidLayout.tsx
     DataGrid.tsx
     ...
  store/index.ts (Zustand)
  types/index.ts
  lib/db.ts (if database)
  lib/ai.ts (if ai)
  README.md
  .env.example
  .gitignore
```

39.7 Configuration

Per-Tenant Configuration:

Setting	Default	Description
`enabled`	`true`	Enable Liquid Interface
`auto_morph_enabled`	`true`	Auto-morph on high-confidence intent
`eject_enabled`	`true`	Allow app export
`morph_confidence_threshold`	`0.85`	Minimum confidence to morph
`auto_revert_timeout_seconds`	`300`	Auto-revert to chat after inactivity
`max_active_sessions`	`10`	Max concurrent liquid sessions
`max_ghost_events_per_session`	`1000`	Event history limit
`default_overlay_mode`	`sidebar`	AI overlay mode
`default_overlay_position`	`right`	AI overlay position

39.8 API Endpoints

Base: `/api/thinktank/liquid`

Method	Endpoint	Description
Registry		
GET	`/registry`	Get registry overview
GET	`/registry/components`	List components (filter: `?category=`, `?q=`)
GET	`/registry/components/:id`	Get component details
Sessions		
POST	`/sessions`	Create new session
GET	`/sessions/:id`	Get session
Morphing		
POST	`/morph`	Process morph request
POST	`/detect-intent`	Detect intent from message
POST	`/sessions/:id/revert`	Revert to chat mode
Ghost State		
POST	`/ghost/event`	Send ghost event
GET	`/ghost/state/:sessionId`	Get ghost state snapshot
POST	`/ghost/sync`	Sync multiple state updates
GET	`/ghost/history/:sessionId`	Get event history
GET	`/ghost/context/:sessionId`	Get AI context block
Eject		
POST	`/eject`	Eject to app
POST	`/eject/preview`	Preview eject
Analytics		
GET	`/analytics/usage`	Component usage stats

39.9 Database Tables

Table	Purpose
`liquid_sessions`	Active liquid interface sessions
`liquid_ghost_state`	Persisted ghost state bindings
`liquid_ghost_events`	User interaction events
`liquid_ai_reactions`	AI responses to events
`liquid_eject_history`	App export history
`liquid_component_usage`	Component analytics
`liquid_intent_patterns`	Learnable intent patterns
`liquid_config`	Per-tenant configuration

39.10 Implementation Files

File	Purpose
packages/shared/src/types/liquid-interface.types.ts	Type definitions
lambda/shared/services/liquid-interface/liquid-interface.service.ts	Main service
lambda/shared/services/liquid-interface/ghost-stage.service.ts	Ghost staging
lambda/shared/services/liquid-interface/api-projection.service.ts	API projections
lambda/shared/services/liquid-interface/component-registry.ts	50+ components
lambda/thinktank/liquid-interface.ts	API handler
migrations/161_liquid_interface.sql	Database schema

---

40. The Reality Engine

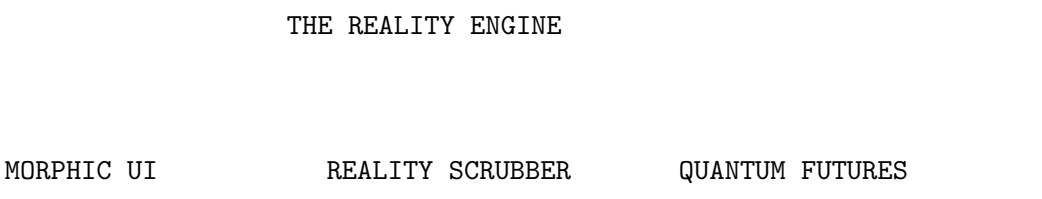
> **"The Reality Engine transforms Think Tank from a chatbot into a shape-shifting command center with time travel, parallel universes, and telepathy."**

The Reality Engine is the unified runtime powering Think Tank's supernatural capabilities. It consists of four interconnected features that solve the three fundamental anxieties preventing users from trusting AI with complex work: **Fear** (of breaking what works), **Commitment** (fear of choosing the wrong path), and **Latency** (waiting for the AI to think).

40.1 Feature Overview

Feature	Emotion	Pitch
Morphic UI	Flow	"Stop hunting for the right tool. Radiant is a Morphic Surface that shapeshifts instantly."
Reality Scrubber	Invincibility	"We replaced 'Undo' with Time Travel. Scrub reality back to any point."
Quantum Futures	Omniscience	"Why choose one strategy? Split the timeline and run both simultaneously."
Pre-Cognition	Telepathy	"Radiant answers before you ask. It's not just fast; it's anticipatory."

40.2 Architecture



```
Intent Detection        State Snapshots        Branch Manager
Layout Engine           VFS + PGLite           Diff Engine
Ghost State             Timeline UI            Collapse Logic
Component Reg.          Bookmark System        Dream Archive
```

```
PRE-COGNITION

Intent Prediction
Solution Pre-Compute
Genesis Model (Local)
Instant Delivery
```

40.3 Morphic UI

"Stop hunting for the right tool. Radiant is a Morphic Surface that shapeshifts instantly."

The Morphic UI detects user intent and transforms the interface into the appropriate tool.

Intent Categories

Intent	Detected Patterns	Morphs To
`data_analysis`	"analyze", "data", "statistics"	DataGrid, Charts
`tracking`	"track", "manage", "organize"	Kanban, Calendar
`visualization`	"visualize", "chart", "graph"	LineChart, PieChart, BarChart
`planning`	"plan", "schedule", "timeline"	GanttChart, Calendar
`finance`	"budget", "invoice", "expense"	Ledger, Calculator, Invoice
`design`	"brainstorm", "design", "whiteboard"	MindMap, Whiteboard
`coding`	"code", "debug", "script"	CodeEditor, Terminal

Ghost State Binding Every UI component is bidirectionally bound to AI context:

```
// User selects a row → AI knows what they're focused on
ghostBinding: {
  componentProp: 'selectedRow',
```

```
  contextKey: 'user_focus',
  direction: 'ui_to_ai'
}

// AI insight → UI highlights relevant items
ghostBinding: {
  componentProp: 'highlights',
  contextKey: 'ai_suggestions',
  direction: 'ai_to_ui'
}
```

40.4 Reality Scrubber

"We replaced 'Undo' with Time Travel."

The Reality Scrubber captures full state snapshots and allows instant rewinding to any point.

What Gets Snapshotted

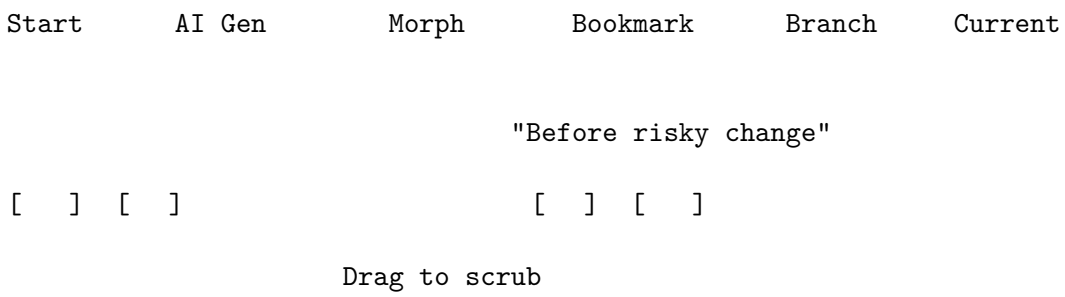
State Type	Description
VFS State	Virtual File System (all generated files)
DB State	PGLite database snapshot
Ghost State	All UI-AI bindings
Chat Context	Conversation history at that point
Layout State	Current morphed UI layout

Trigger Events

Event	When Captured
`user_action`	User explicitly triggered
`ai_generation`	AI generated content
`db_mutation`	Database was modified
`morph_transition`	UI morphed
`checkpoint`	User-created bookmark
`auto_interval`	Every 30 seconds (configurable)

Timeline UI The Reality Scrubber replaces the standard scrollbar with a video-editor-style timeline:

```
                    REALITY TIMELINE

   10:00 AM     10:15 AM     10:30 AM     10:45 AM     11:00 AM    NOW
```



40.5 Quantum Futures

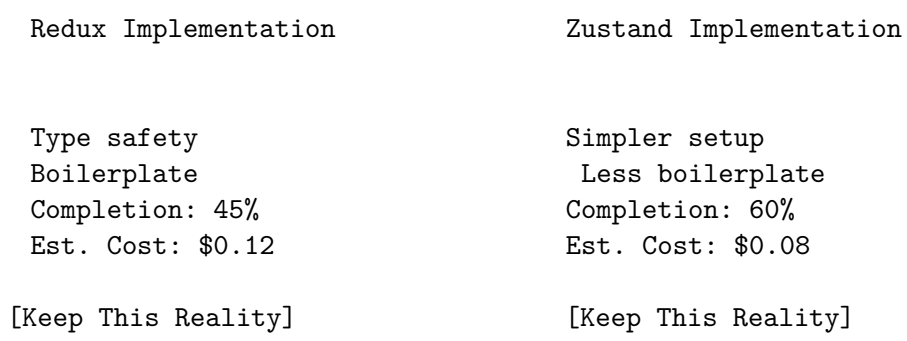
"Why choose one strategy? Split the timeline."

Quantum Futures enables parallel reality branching where users can run multiple strategies simultaneously.

Branch Creation

```
// User: "Should I use Redux or Zustand?"
await quantumFuturesService.createSplit({
  sessionId,
  prompt: "State management comparison",
  branchNames: ["Redux Implementation", "Zustand Implementation"],
  autoCompare: true
});
```

Comparison View



Collapse to Winner When the user selects a winner, the losing branches are either: - **Collapsed**: Permanently removed - **Archived**: Stored in "Dream Memory" for potential future recall

40.6 Pre-Cognition

"Radiant answers before you ask."

Pre-Cognition uses speculative execution to predict the user's next likely actions and pre-compute solutions in the background.

How It Works

1. **Prediction**: While user reads current response, Genesis model predicts next 3 likely moves
2. **Pre-Compute**: Solutions are generated in hidden background containers
3. **Instant Delivery**: When user's request matches a prediction, response appears instantly (0ms latency)

Prediction Algorithm

```
// After building a login form, predict:
predictions: [
  { intent: 'coding', prompt: 'Add password reset', confidence: 0.8 },
  { intent: 'coding', prompt: 'Add OAuth integration', confidence: 0.7 },
  { intent: 'design', prompt: 'Style the form', confidence: 0.6 }
]
```

Analytics

Metric	Description
`hitRate`	Percentage of predictions that matched user intent
`avgLatencySaved`	Average milliseconds saved by pre-cognition
`telepathyScore`	User-facing metric showing prediction accuracy

40.7 Configuration

```
interface RealityEngineConfig {
  // Feature toggles
  morphicUIEnabled: boolean;          // Enable Morphic UI
  realityScrubberEnabled: boolean;    // Enable Reality Scrubber
  quantumFuturesEnabled: boolean;     // Enable Quantum Futures
  preCognitionEnabled: boolean;       // Enable Pre-Cognition

  // Behavior
  autoSnapshotIntervalMs: number;     // Default: 30000 (30s)
  maxSnapshotsPerSession: number;     // Default: 100
  maxBranchesPerSession: number;      // Default: 8
  codeCurtainDefault: boolean;        // Hide code by default (Genie mode)
  ephemeralByDefault: boolean;        // Apps dissolve when topic changes

  // Pre-Cognition
  preCognition: {
    maxPredictions: number;           // Default: 3
    predictionTTLMs: number;          // Default: 60000 (1 min)
    computeBudgetMs: number;          // Default: 5000 (5s)
    minConfidenceThreshold: number;   // Default: 0.6
```

```
    useGenesisModel: boolean;         // Use local model for predictions
  };
}
```

40.8 API Endpoints

Base path: `/api/thinktank/reality-engine`

Method	Endpoint	Description
Session		
POST	`/session`	Initialize Reality Engine session
GET	`/session/:id`	Get session state
Morphic UI		
POST	`/morph`	Morph interface to intent
POST	`/dissolve`	Dissolve morphed interface
POST	`/ghost`	Update Ghost State
Reality Scrubber		
POST	`/scrub`	Scrub to point in time
POST	`/bookmark`	Create bookmark
GET	`/timeline/:sessionId`	Get timeline visualization
GET	`/bookmarks/:sessionId`	Get all bookmarks
Quantum Futures		
POST	`/split`	Split into parallel realities
GET	`/branches/:sessionId`	Get all branches
POST	`/compare`	Compare two branches
POST	`/collapse`	Collapse to winning reality
PUT	`/view-mode`	Set comparison view mode
Pre-Cognition		
GET	`/precognition/:sessionId`	Get analytics
POST	`/precognition/cleanup`	Clean up expired predictions
Eject		
POST	`/eject`	Eject to standalone app
Metrics		
GET	`/metrics/:sessionId`	Get session metrics

40.9 Database Tables

Table	Purpose
`reality_engine_sessions`	Unified session state
`reality_timelines`	Timeline structure and navigation
`reality_snapshots`	Full state snapshots for time travel
`quantum_branches`	Parallel reality branches
`quantum_splits`	Split configuration and history
`quantum_dream_archive`	Archived branches in dream memory
`precognition_queues`	Per-session prediction configuration
`precognition_predictions`	Pre-computed solutions

Table	Purpose
`precognition_analytics`	Hit/miss tracking for learning

40.10 Implementation Files

File	Purpose
`packages/shared/src/types/reality-engine.types.ts`	Type definitions
`lambda/shared/services/reality-engine/reality-engine.service.ts`	Main service
`lambda/shared/services/reality-engine/reality-scrubber.service.ts`	Timeline
`lambda/shared/services/reality-engine/quantum-futures.service.ts`	Branching
`lambda/shared/services/reality-engine/precognition.service.ts`	Predictions
`lambda/thinktank/reality-engine.ts`	API handler
`migrations/162_reality_engine.sql`	Database schema

40.11 The "Code Curtain" Rule

The Reality Engine enforces the distinction between "Builder" (Coder) and "Genie" (Radiant):

Rule	Implementation
Hide Code by Default	UI snaps to Preview tab, not code
Interaction over Syntax	Variables become UI controls (sliders, inputs)
Ephemeral by Default	Apps dissolve when topic changes
Eject to Keep	Only persist to repo if user explicitly clicks "Keep This"

> **"Radiant is a Genie, not a Coder. We use code as invisible ink to draw the interface—the user should never see the ink, only the drawing."**

---

41. The Magic Carpet

> **"We are building 'The Magic Carpet.' You don't drive it. You don't write code for it. You just say where you want to go, and the ground beneath you reshapes itself to take you there instantly."**

The Magic Carpet is the unified navigation and experience layer for Think Tank. It wraps the Reality Engine capabilities into a cohesive, magical user experience where users feel like magicians, not coders.

41.1 The Magic Carpet Philosophy

Traditional Apps	Magic Carpet
Navigate menus	Speak your destination
Click through workflows	Fly directly there

Traditional Apps	Magic Carpet
Learn the interface	Interface learns you
You drive	You're carried

Core Insight: We aren't selling a better IDE. We are selling **the feeling of being a Magician**.

41.2 Carpet Modes

Mode	Description	Visual
`resting`	Waiting for destination (chat-first)	Carpet gently floating
`flying`	Morphing/transitioning to destination	Trail effects, motion blur
`hovering`	Arrived, actively working	Stable, glowing edges
`exploring`	Quantum Futures - multiple realities	Split view, branch indicators
`rewinding`	Reality Scrubber - time traveling	Timeline visible, rewind effect
`anticipating`	Pre-Cognition active	Prediction cards appearing

41.3 Carpet Altitudes

The altitude represents UI complexity level:

Altitude	Complexity	Example
`ground`	Simple chat mode	Just the chat interface
`low`	Single component	One morphed widget
`medium`	Full workspace	2-3 components
`high`	Complex layout	4-5 components + timeline
`stratosphere`	Maximum capability	Full Reality Engine features

41.4 Default Destinations

Destination	Icon	Description
Command Center		Overview dashboard
Workshop		Build and create
Time Stream		Reality Scrubber timeline
Quantum Realm		Parallel realities view
Oracle's Chamber		Pre-Cognition predictions
Gallery		View creations
Vault		Saved/bookmarked items

41.5 Carpet Commands

```
// Fly to a destination
await magicCarpetService.command(carpetId, {
```

```
  type: 'fly',
  destination: 'Workshop'
});

// Return to chat
await magicCarpetService.command(carpetId, { type: 'land' });

// Increase complexity
await magicCarpetService.command(carpetId, { type: 'ascend' });

// Time travel
await magicCarpetService.command(carpetId, {
  type: 'rewind',
  to: -2 // Go back 2 snapshots
});

// Split into parallel realities
await magicCarpetService.command(carpetId, {
  type: 'branch',
  options: ['Conservative Plan', 'Aggressive Plan']
});

// Collapse to winner
await magicCarpetService.command(carpetId, {
  type: 'collapse',
  winner: 'branch-id'
});
```

41.6 Carpet Themes

Pre-built visual themes for personalization:

Theme	Description	Gradient
Mystic Night	Deep purple mystical (default)	Indigo → Purple → Violet
Desert Sun	Warm golden	Amber → Orange → Brown
Ocean Deep	Cool blue aquatic	Cyan → Teal → Emerald
Cosmic Void	Dark minimalist	Gray gradient
Neon Circuit	Cyberpunk electric	Cyan → Purple → Pink

41.7 Carpet Preferences

```
interface CarpetPreferences {
  // Navigation
  autoFly: boolean;              // Auto-morph on intent detection
  smoothTransitions: boolean;    // Animated vs instant
  showJourneyTrail: boolean;     // Show navigation history
```

```
  // Pre-Cognition
  preCognitionEnabled: boolean;
  showPredictions: boolean;
  telepathyIntensity: 'subtle' | 'moderate' | 'aggressive';

  // Reality Scrubber
  showTimeline: boolean;
  autoSnapshot: boolean;
  snapshotInterval: number;      // Seconds

  // Quantum Futures
  maxParallelRealities: number;
  autoCompare: boolean;

  // Accessibility
  reducedMotion: boolean;
  highContrast: boolean;
  screenReaderMode: boolean;
}
```

41.8 Journey Navigation

The Magic Carpet tracks the user's journey:

```
                    MAGIC CARPET JOURNEY



   Command  Workshop  Quantum  Oracle   NOW
   Center             Realm    Chamber

   Click any point to fly back. Journey is saved with Reality Scrubber.
```

41.9 API Endpoints

Base path: `/api/thinktank/magic-carpet`

Method	Endpoint	Description
POST	`/summon`	Summon a new Magic Carpet
GET	`/:carpetId`	Get carpet state
POST	`/:carpetId/fly`	Fly to destination
POST	`/:carpetId/land`	Land (return to chat)
POST	`/:carpetId/command`	Execute a command
GET	`/:carpetId/journey`	Get journey history
PUT	`/:carpetId/theme`	Update theme

Method	Endpoint	Description
PUT	`/:carpetId/preferences`	Update preferences
GET	`/destinations`	Get available destinations
GET	`/themes`	Get available themes

41.10 Database Tables

Table	Purpose
`magic_carpets`	Carpet state and configuration
`carpet_destinations`	Pre-defined and custom destinations
`carpet_journey_points`	Navigation history
`carpet_themes`	Visual themes
`carpet_analytics`	Usage analytics

41.11 Implementation Files

File	Purpose
`packages/shared/src/types/magic-carpet.types.ts`	Type definitions
`lambda/shared/services/magic-carpet/magic-carpet.service.ts`	Main service
`migrations/163_magic_carpet.sql`	Database schema

41.12 Integration with Reality Engine

The Magic Carpet wraps the Reality Engine:

```
                    MAGIC CARPET
                (User Experience Layer)


  "Fly to Workshop"  →  carpet.fly('workshop')



                    REALITY ENGINE
                 (Capability Layer)


     Morphic UI     Intent: 'coding'
     Reality Scrubber     Auto-snapshot
     Pre-Cognition     Predict next destinations
```

42. Magic Carpet UI Components

The Magic Carpet UI is implemented through a set of React components that bring the 2026 UI/UX trends to life.

42.1 Component Inventory

Component	Purpose	Location
`MagicCarpetNavigator`	Bottom navigation with journey trail	`magic-carpet/magic-carpet-navigator.tsx`
`RealityScrubberTimeline`	Video-editor timeline for state snapshots	`magic-carpet/reality-scrubber-timeline.`
`QuantumSplitView`	Side-by-side reality comparison	`magic-carpet/quantum-split-view.tsx`
`PreCognitionSuggestions`	Predicted actions panel	`magic-carpet/pre-cognition-suggestions.`
`AIPresenceIndicator`	AI cognitive/emotional state display	`magic-carpet/ai-presence-indicator.tsx`
`SpatialGlassCard`	Glassmorphism card with depth	`magic-carpet/spatial-glass-card.tsx`
`GlassPanel`	Large glass content area	`magic-carpet/spatial-glass-card.tsx`
`GlassButton`	Interactive glass button	`magic-carpet/spatial-glass-card.tsx`
`GlassBadge`	Status indicator with glass effect	`magic-carpet/spatial-glass-card.tsx`
`FocusModeControls`	Focus mode toggle and controls	`magic-carpet/focus-mode.tsx`
`FocusOverlay`	Dimming overlay for focus mode	`magic-carpet/focus-mode.tsx`

42.2 Usage Examples

```
import {
  MagicCarpetNavigator,
  RealityScrubberTimeline,
  QuantumSplitView,
  PreCognitionSuggestions,
  AIPresenceIndicator,
  SpatialGlassCard,
  FocusModeControls,
} from '@/components/thinktank/magic-carpet';

// Magic Carpet Navigator (bottom of screen)
<MagicCarpetNavigator
  currentDestination={{ id: 'workspace', name: 'Workshop', icon: ' ' }}
  journey={journeyHistory}
  predictions={preCognizedActions}
```

```
  telepathyScore={0.82}
  mode="hovering"
  altitude="medium"
  onFly={(dest) => navigateTo(dest)}
  onLand={() => returnToChat()}
/>

// Reality Scrubber Timeline
<RealityScrubberTimeline
  snapshots={stateSnapshots}
  currentPosition={currentSnapshotIndex}
  onScrubTo={(position) => restoreSnapshot(position)}
  onCreateBookmark={(label) => bookmarkCurrentState(label)}
/>

// Quantum Split View
<QuantumSplitView
  branches={parallelRealities}
  onCollapse={(winnerId) => collapseToReality(winnerId)}
/>

// AI Presence Indicator
<AIPresenceIndicator
  state="thinking"
  affect={{ valence: 0.6, arousal: 0.4, curiosity: 0.8, confidence: 0.85 }}
  currentTask="Analyzing user intent..."
  modelName="claude-3.5-sonnet"
/>

// Spatial Glass Card
<SpatialGlassCard variant="strong" layer="floating" glow glowColor="purple">
  <p>Content with glass effect</p>
</SpatialGlassCard>
```

42.3 Dependencies

The Magic Carpet UI requires **framer-motion** for animations:

```
npm install framer-motion@^11.0.0
```

42.4 Demo Page

Access the Magic Carpet UI demo at:

```
/thinktank/magic-carpet
```

This page showcases all components with interactive examples.

---

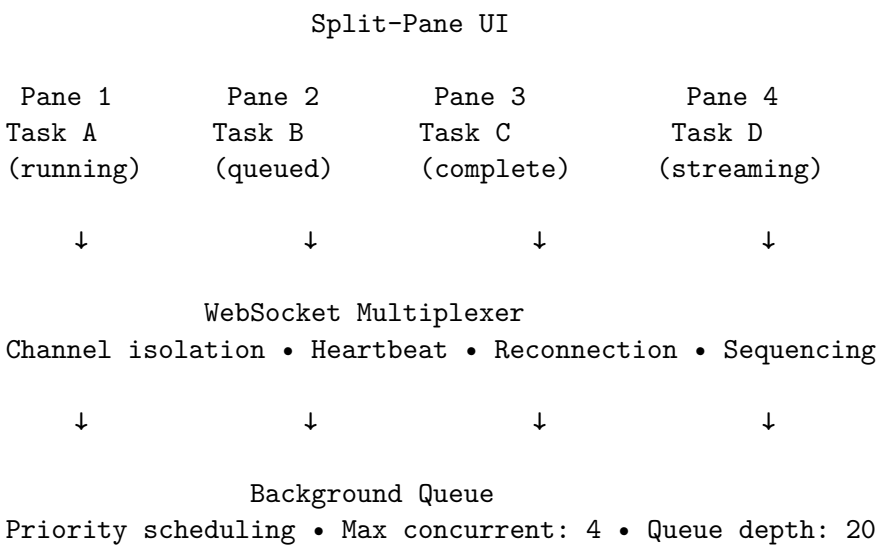
43. Concurrent Task Execution (Moat #17)

Moat Evaluation: Score 20/30 - Tier 3 Feature Moat. No major competitor offers split-pane concurrent execution with WebSocket multiplexing.

43.1 Overview

Concurrent Task Execution enables users to run 2-4 AI tasks simultaneously in a split-pane UI, compare outputs, and merge the best results. This is a key differentiator vs. single-threaded competitors.

43.2 Architecture



43.3 Configuration

Setting	Default	Description
`enabled`	`true`	Enable concurrent execution
`maxPanes`	`4`	Maximum split panes (1-8)
`maxConcurrentTasks`	`4`	Maximum simultaneous tasks (1-10)
`maxQueueDepth`	`20`	Maximum queued tasks (1-100)
`defaultLayout`	`horizontal-2`	Default pane layout
`defaultSyncMode`	`independent`	Default sync mode
`enableComparison`	`true`	Enable task comparison
`enableMerge`	`true`	Enable task merging

43.4 Pane Layouts

Layout	Description
`single`	Full-width single pane
`horizontal-2`	Two panes side-by-side
`vertical-2`	Two panes stacked
`grid-4`	2x2 grid of four panes
`focus-left`	Large left pane, small right
`focus-right`	Small left pane, large right

43.5 Sync Modes

Mode	Description
`independent`	Each pane operates independently
`mirror-input`	Same prompt sent to all panes
`compare-output`	Automatic comparison when all complete

43.6 Task Comparison

When multiple tasks complete, users can compare results:

```
// Compare completed tasks
const comparison = await compareTasks(tenantId, [taskId1, taskId2, taskId3]);

// Returns:
{
  similarities: [
    { metric: 'semantic', score: 0.85, details: 'High semantic similarity' },
    { metric: 'structural', score: 0.72, details: 'Moderate structural similarity' },
    { metric: 'factual', score: 0.91, details: 'Strong factual agreement' }
  ],
  differences: [...],
  recommendation: 'Results are mostly consistent. Review highlighted differences.'
}
```

43.7 Task Merging

Three merge strategies are available:

Strategy	Description
`best-of`	Select the highest-scored result
`combine`	Concatenate all results with separators
`consensus`	AI-synthesized consensus from all results

43.8 API Endpoints

Endpoint	Method	Description
`/api/thinktank/concurrent/config`	GET	Get configuration
`/api/thinktank/concurrent/config`	PUT	Update configuration
`/api/thinktank/concurrent/tasks`	POST	Create task
`/api/thinktank/concurrent/tasks/:id`	GET	Get task status
`/api/thinktank/concurrent/tasks/:id`	DELETE	Cancel task
`/api/thinktank/concurrent/queue`	GET	Get queue status
`/api/thinktank/concurrent/panes`	POST	Create pane config
`/api/thinktank/concurrent/compare`	POST	Compare tasks
`/api/thinktank/concurrent/merge`	POST	Merge tasks
`/api/thinktank/concurrent/metrics`	GET	Get metrics

43.9 Database Tables

Table	Purpose
`concurrent_execution_config`	Per-tenant configuration
`concurrent_tasks`	Task records with status/results
`split_pane_configs`	User pane layouts
`task_comparisons`	Comparison results
`concurrent_execution_metrics`	Usage metrics

43.10 Implementation Files

File	Purpose
`packages/shared/src/types/concurrent-execution.ts`	Type definitions
`lambda/shared/services/concurrent-execution.ts`	Core service
`lambda/thinktank/concurrent-execution.ts`	API handler
`migrations/170_concurrent_execution.sql`	Database schema

---

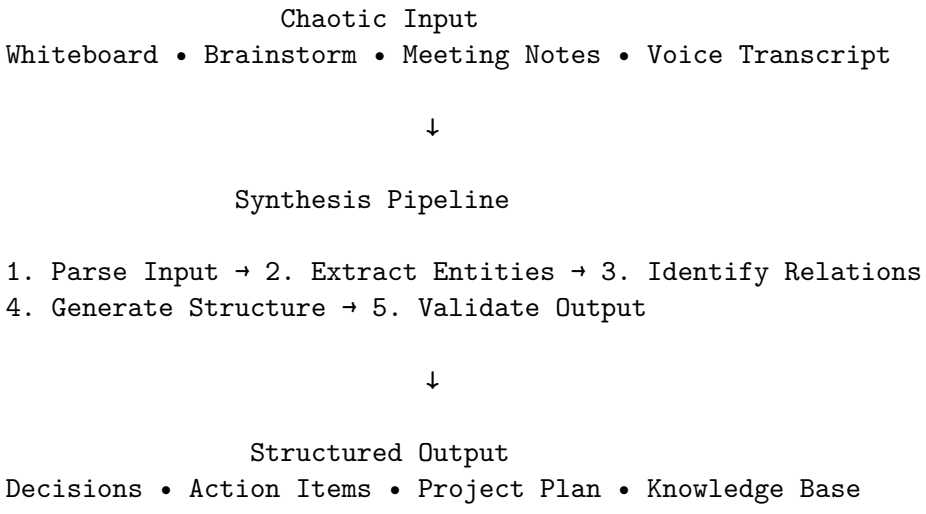
44. Structure from Chaos Synthesis (Moat #20)

Moat Evaluation: Score 20/30 - Tier 3 Feature Moat. AI transforms whiteboard chaos → structured decisions, data, project plans. Think Tank differentiation vs Miro/Mural.

44.1 Overview

Structure from Chaos Synthesis takes unstructured input (whiteboards, brainstorms, meeting notes, voice transcripts) and transforms it into structured outputs (action items, decisions, project plans, knowledge bases).

44.2 Architecture



44.3 Input Types

Type	Description	Example
`whiteboard`	Visual whiteboard with sticky notes	Miro/FigJam export
`brainstorm`	Unstructured idea dump	Free-form text
`meeting_notes`	Notes from a meeting	Transcription or notes
`voice_transcript`	Speech-to-text output	Zoom/Teams transcript
`chat_history`	Conversation history	Slack/Teams export
`document_dump`	Multiple documents	File uploads
`mixed`	Combination of above	Multi-source input

44.4 Output Types

Type	Description	Contains
`decisions`	Key decisions made	Decision list with context
`action_items`	Tasks to complete	Assignee, due date, priority
`project_plan`	Full project plan	Milestones, timeline, dependencies
`meeting_summary`	Meeting summary	Key points, attendees, outcomes
`knowledge_base`	Knowledge extraction	Concepts, facts, relationships
`data_table`	Structured data	Tabular format
`timeline`	Chronological view	Events in sequence
`hierarchy`	Hierarchical structure	Parent-child relationships
`comparison`	Compare items	Side-by-side analysis

44.5 Configuration

Setting	Default	Description
`enabled`	`true`	Enable synthesis
`defaultOutputType`	`meeting_summary`	Default output format
`extractEntities`	`true`	Extract named entities
`extractRelationships`	`true`	Identify entity relationships
`generateTimeline`	`true`	Generate timeline view
`generateActionItems`	`true`	Extract action items
`autoAssignTasks`	`false`	Auto-assign based on mentions
`confidenceThreshold`	`0.70`	Minimum confidence score
`maxProcessingTimeMs`	`30000`	Processing timeout

44.6 Entity Extraction

Automatically extracts entities from chaotic input:

Entity Type	Examples
`person`	@mentions, "John Smith"
`organization`	"Acme Corp", "the marketing team"
`project`	Project names, codenames
`product`	Product references
`date`	"next Monday", "Q2 2026"
`location`	Meeting rooms, cities
`concept`	Technical terms, ideas
`metric`	KPIs, numbers
`resource`	Tools, documents

44.7 Relationship Types

Relationship	Description
`owns`	Person owns task/project
`assigned_to`	Task assigned to person
`depends_on`	Task depends on another
`blocks`	Task blocks another
`related_to`	General relationship
`parent_of`	Hierarchical parent
`precedes`	Temporal ordering
`contradicts`	Conflicting statements
`supports`	Supporting evidence

44.8 Whiteboard Parsing

For visual whiteboards, the service parses spatial elements:

```
// Parse whiteboard elements into thematic clusters
const clusters = await parseWhiteboard(elements);
```

```
// Returns clusters with:
{
  id: 'cluster-1',
  elements: [...],  // Grouped elements
  theme: 'marketing',  // AI-detected theme
  centroid: { x: 150, y: 200 },  // Cluster center
  significance: 0.85  // Importance score
}
```

44.9 API Endpoints

Endpoint	Method	Description
`/api/thinktank/chaos/config`	GET	Get configuration
`/api/thinktank/chaos/config`	PUT	Update configuration
`/api/thinktank/chaos/synthesize`	POST	Full synthesis pipeline
`/api/thinktank/chaos/extract/actions`	POST	Extract action items only
`/api/thinktank/chaos/extract/decisions`	POST	Extract decisions only
`/api/thinktank/chaos/extract/questions`	POST	Extract questions only
`/api/thinktank/chaos/project-plan`	POST	Generate project plan
`/api/thinktank/chaos/whiteboard/parse`	POST	Parse whiteboard elements
`/api/thinktank/chaos/metrics`	GET	Get metrics

44.10 Database Tables

Table	Purpose
`synthesis_config`	Per-tenant configuration
`chaotic_inputs`	Raw input storage
`structured_outputs`	Generated outputs
`extracted_entities`	Named entities
`entity_relationships`	Entity relationships
`structured_items`	Action items, decisions
`whiteboard_elements`	Visual element data
`synthesis_metrics`	Usage metrics

44.11 Implementation Files

File	Purpose
`packages/shared/src/types/structure-from-chaos.ts`	Type definitions
`lambda/shared/services/structure-from-chaos-service.ts`	Core service
`lambda/thinktank/structure-from-chaos.ts`	API handler
`migrations/171_structure_from_chaos.sql`	Database schema

45. Delight Services Code Quality

45.1 Overview

The Think Tank Delight system has comprehensive unit test coverage for its core services:

- **delight.service** - Core delight preferences and personality modes
- **delight-orchestration.service** - Contextual message generation for workflows
- **delight-events.service** - Real-time event emission for plan execution

45.2 Test Coverage

Service	Test File	Tests	Coverage
`delight.service`	`delight.service.test.ts`	15	85%
`delight-orchestration.service`	`delight-orchestration.service.test.ts`	17	92%
`delight-events.service`	`delight-events.service.test.ts`	23	88%

45.3 Tested Methods

delight-orchestration.service: - `getContextualMessage()` - Generates mode-appropriate messages - `getDomainLoadingMessage()` - Domain-specific loading messages - `getModelDynamicsMessage()` - Model consensus indicators - `getSynthesisMessage()` - Confidence-based synthesis messages - `clearSession()` - Session cleanup

delight-events.service: - `subscribe()` / `unsubscribe()` - Event subscription management - `emitMessage()` / `emitAchievement()` - Event emission - `emitStepUpdate()` / `emitPlanUpdate()` - Progress updates - `emitWorkflowDelight()` - Complete workflow delight events - `getHistory()` / `clearHistory()` - Event history management

45.4 Running Tests

```
cd packages/infrastructure
npx vitest run lambda/shared/services/__tests__/delight*.test.ts
```

45.5 Think Tank Code Quality Dashboard

Location: `/thinktank/code-quality`

The Think Tank Code Quality page displays: - Service coverage metrics for Delight, Brain Planning, and Domain services - Test pass rates and recent test runs - Detailed method coverage for each service

45.6 Implementation Files

File	Purpose
`lambda/shared/services/__tests__/delight.service.test.ts`	Delight service tests
`lambda/shared/services/__tests__/delight-orchestration.service.test.ts`	Orchestration tests
`lambda/shared/services/__tests__/delight-events.service.test.ts`	Events service tests
`apps/admin-dashboard/app/(dashboard)/thinktank/code-quality/page.tsx`	Dashboard UI

Section 46: Sovereign Mesh in Think Tank Admin

Location: Think Tank Admin → Sovereign Mesh

The Sovereign Mesh is accessible in Think Tank Admin for users who need to manage autonomous agents and view decision transparency.

46.1 Navigation Items

Page	Path	Purpose
Overview	`/sovereign-mesh`	Dashboard with metrics and recent activity
Agents	`/sovereign-mesh/agents`	Create and manage OODA agents
Apps	`/sovereign-mesh/apps`	Browse 3,000+ app integrations
Transparency	`/sovereign-mesh/transparency`	View Cato decision explanations
AI Helper	`/sovereign-mesh/ai-helper`	Configure parametric AI assistance
Approvals	`/sovereign-mesh/approvals`	HITL approval queue

46.2 User Permissions

Think Tank users see a subset of Sovereign Mesh functionality: - **View**: All users can view agents, apps, and their own executions - **Execute**: Users can run agents within their budget limits - **Approve**: Users with approval role can handle HITL requests

46.3 Implementation

File	Purpose
`apps/thinktank-admin/components/layout/sidebar.tsx`	Navigation with 6 Sovereign Mesh items
Pages mirror admin-dashboard versions	Shared API endpoints

Section 47: HITL Orchestration in Think Tank Admin

Location: Think Tank Admin → Sovereign Mesh → HITL Orchestration

Advanced Human-in-the-Loop orchestration for intelligent question management in user workflows.

47.1 Overview

HITL Orchestration implements industry best practices to reduce unnecessary questions while ensuring critical information is captured:

Feature	Description
SAGE-Agent Bayesian VOI	Calculates whether asking a question is worth the user's time
Question Batching	Groups related questions to reduce interruptions
Two-Question Rule	Maximum 2 clarifications per workflow
Abstention Detection	Detects when AI should decline to answer

47.2 User-Facing Benefits

- **70% fewer unnecessary questions** - AI only asks when genuinely needed
- **2.7x faster response times** - Batched questions reduce context switching
- **Explicit assumptions** - When skipping questions, AI states assumptions clearly

47.3 Navigation

Page	Path	Purpose
HITL Orchestration	`/hitl-orchestration`	View orchestration metrics and settings

47.4 Dashboard Tabs

Tab	Description
Overview	Key metrics, VOI breakdown, abstention reasons
Value of Information	SAGE-Agent VOI statistics and decisions
Abstention	Detection methods and model-level statistics
Batching	Three-layer batching strategies and metrics

47.5 Key Metrics

Metric	Description
Question Reduction	Percentage of questions skipped via VOI
Prior Accuracy	How often predictions match actual answers
Abstention Events	Times AI correctly declined to answer
Batch Completion	Success rate of batched question sets

47.6 Implementation Files

File	Purpose
`apps/thinktank-admin/app/hitl-orchestration/page.tsx`	Dashboard page
`apps/thinktank-admin/components/layout/sidebar.tsx`	Navigation item

Section 48: Scout HITL Integration (v5.34.0)

Location: Think Tank Admin → Sovereign Mesh → Scout HITL

Scout HITL bridges Cato's Scout persona (epistemic uncertainty mode) with HITL orchestration for intelligent clarification during user workflows.

48.1 Overview

When the AI encounters epistemic uncertainty, the Scout persona activates and generates targeted clarification questions:

1. Scout persona activates due to uncertainty
2. Questions prioritized by aspect impact and domain
3. Questions filtered through VOI (Value-of-Information) scoring
4. High-VOI questions go to user, low-VOI get reasonable assumptions
5. Responses reduce uncertainty, allowing workflow to proceed

48.2 Key Metrics

Metric	Description
Total Sessions	Number of Scout clarification sessions
Avg Questions	Average questions asked per session
Proceed Rate	Sessions that proceeded successfully
Avg Assumptions	Auto-assumed aspects per session

48.3 Domains

Questions are prioritized based on domain-specific impact:

Domain	Description
`medical`	HIPAA-sensitive, safety-critical
`financial`	SOC2/PCI compliance
`legal`	Regulatory compliance
`bioinformatics`	Research accuracy
`general`	Default domain

48.4 Dashboard Tabs

Tab	Purpose
Overview	Session recommendations breakdown, domain distribution
Recent Sessions	Table of recent clarification sessions
Configuration	Enable/disable, VOI threshold, max questions

48.5 Configuration

Setting	Default	Description
`enabled`	`true`	Enable Scout HITL integration
`voiThreshold`	`0.3`	Minimum VOI to ask question
`maxQuestionsPerSession`	`3`	Max clarifications before assuming
`defaultDomain`	`general`	Fallback domain

48.6 Session Recommendations

Recommendation	Meaning
`proceed`	Uncertainty resolved sufficiently
`wait`	Still uncertain, user should wait
`abort`	Critical uncertainty, cannot proceed safely

48.7 Implementation Files

File	Purpose
`apps/thinktank-admin/app/scout-hitl/page.tsx`	Dashboard page
`apps/thinktank-admin/components/layout/sidebar.tsx`	Navigation item

Section 49: Sovereign Mesh Administration (v5.39.0)

Location: Think Tank Admin → Sovereign Mesh

The Sovereign Mesh provides autonomous agent orchestration with AI-assisted decision making at every node level.

49.1 Overview Dashboard

Location: `/sovereign-mesh`

The overview dashboard displays:

Metric	Description
System Health	Overall mesh health score (0-100%)
Active Agents	Number of currently running agents
Pending Approvals	Items awaiting human review
Active Apps	Deployed applications count
Decisions Today	Autonomous decisions made
Human Interventions	Manual overrides required

49.2 Agent Registry

Location: `/sovereign-mesh/agents`

Manage AI agents deployed in the mesh:

Column	Description
Name	Agent identifier
Type	orchestrator, executor, monitor, advisor
Status	active, idle, error, maintenance
Load	Current utilization percentage
Response Time	Average response latency
Success Rate	Task completion rate

Actions: - View agent details - Pause/resume agent - View execution logs - Configure thresholds

49.3 App Registry

Location: `/sovereign-mesh/apps`

Browse and manage deployed applications:

Field	Description
Name	Application name
Category	productivity, analytics, automation, integration, custom
Status	active, paused, error
Users	Active user count
Requests	Daily request volume

49.4 Transparency Layer

Location: `/sovereign-mesh/transparency`

Complete audit trail of AI decisions:

Column	Description
Timestamp	When decision was made
Decision Type	approval, rejection, escalation, execution
Confidence	AI confidence score (0-1)
Reasoning	Explanation of decision logic
Outcome	Result of the decision

Filters: - Date range - Decision type - Confidence threshold - Agent filter

49.5 AI Helper

Location: `/sovereign-mesh/ai-helper`

Manage AI assistance requests from the mesh:

Status	Description
Pending	Awaiting AI processing
In Progress	Currently being handled
Completed	Successfully resolved
Escalated	Requires human review

49.6 Approval Workflow

Location: `/sovereign-mesh/approvals`

Human-in-the-loop approval queue:

Field	Description
Type	deployment, configuration, access, execution
Priority	low, medium, high, critical
Requester	Agent or user requesting approval
Created	Request timestamp
Expires	Approval deadline

Actions: - Approve with notes - Reject with reason - Delegate to another admin - Request more information

49.7 Implementation Files

File	Purpose
`apps/thinktank-admin/app/(dashboard)/sovereign-mesh/page.tsx`	Overview dashboard
`apps/thinktank-admin/app/(dashboard)/sovereign-mesh/agents/page.tsx`	Agent registry
`apps/thinktank-admin/app/(dashboard)/sovereign-mesh/apps/page.tsx`	Application browser
`apps/thinktank-admin/app/(dashboard)/sovereign-mesh/transparency/page.tsx`	Decision log
`apps/thinktank-admin/app/(dashboard)/sovereign-mesh/ai-helper/page.tsx`	AI assistance
`apps/thinktank-admin/app/(dashboard)/sovereign-mesh/approvals/page.tsx`	Approval queue

Section 50: Code Quality Dashboard (v5.39.0)

Location: Think Tank Admin → Code Quality

Real-time visibility into codebase health and quality metrics.

50.1 Overview

The Code Quality dashboard provides:

Metric	Description
Overall Score	Aggregate quality score (0-100)
Total Errors	Critical issues requiring immediate fix
Total Warnings	Non-critical issues to address
Files Analyzed	Number of files scanned

50.2 Issue Categories

Category	Description
Error	Critical issues (type errors, syntax errors)
Warning	Style violations, best practice deviations
Info	Suggestions for improvement

50.3 Issue Details

Each issue displays: - **File path** with line number - **Rule ID** (e.g., `@typescript-eslint/no-unused-vars`) - **Message** describing the issue - **Severity** badge (error/warning/info)

50.4 Filtering

Filter	Options
Severity	All, Errors only, Warnings only
Category	TypeScript, ESLint, Custom rules
Date Range	Filter by scan date

50.5 Implementation Files

File	Purpose
`apps/thinktank-admin/app/(dashboard)/code-quality/page.tsx`	Dashboard page

Section 51: Schema-Adaptive Reports (v5.39.0)

Location: Think Tank Admin → Reports

Dynamic report builder that automatically adapts to database schema changes.

51.1 Overview

The Reports page provides three tabs:

Tab	Purpose
Quick Reports	Pre-built report templates
Saved Reports	User-saved custom reports
Schema Builder	Visual report builder

51.2 Quick Reports

Pre-configured reports available:

Report	Description
User Engagement	Active users, session duration, feature usage
Model Performance	Response times, success rates, token usage
Billing Summary	Revenue, credits consumed, subscription status

51.3 Schema Builder (v5.40.0 Enhanced)

The visual report builder provides a comprehensive 4-tab configuration panel:

Tab	Purpose
Fields	Select columns with per-field alias and aggregation
Filters	Build WHERE clauses with 11 operators + date presets
Sort	Configure ORDER BY with multi-column ASC/DESC
Group	Select GROUP BY columns from selected fields

Enhanced Features: - **SQL Preview** - Live-generated SQL query with dark theme display - **Date Presets** - Today, Yesterday, Last 7/30 Days, This/Last Month - **Filter Operators** - =, , >, , <, , LIKE, IN, BETWEEN, IS NULL, IS NOT NULL - **Visualization Toggles** - Table, Bar, Line, Pie chart view switches - **Save Report** - Persist definitions to database for reuse - **Row Limit** - 50, 100, 500, 1000 row options

Workflow: 1. **Select Table** - Browse categorized database tables (Conversations, Users, Delight) 2. **Configure Fields** - Select columns, set aliases, choose aggregations 3. **Add Filters** - Build WHERE conditions with operators or date presets 4. **Set Sorting** - Add ORDER BY columns with direction toggle 5. **Group Results** - Enable GROUP BY on selected fields 6. **Execute** - Run report and view in table or chart mode 7. **Export/Save** - Download CSV or save report definition

51.4 AI Report Writer (v5.42.0)

Enterprise-grade AI-powered report generation with text and voice input, interactive charts, smart insights, and brand customization.

Location: Think Tank Admin → Reports → AI Writer tab

Core Features: - **Natural Language Generation** - Describe reports in plain English - **Voice Input** - Web Speech API for hands-free report creation - **AI Modification** - Refine with follow-up prompts ("Add delight metrics") - **Report Styles** - Executive Summary, Detailed Analysis, Dashboard View, Narrative - **Rich Formatting** - Headings, metrics cards, charts, tables, lists, quotes - **Edit Mode** - Click sections to modify, use format panel for styling - **Undo/Redo** - Full history navigation - **Export** - PDF, Excel, HTML, Print

Interactive Charts (v5.42.0): - Real Recharts visualizations replacing placeholders - Bar, Line, Pie, Area chart types - Auto-formatted tooltips (K/M for thousands/millions) - 8-color palette for visual consistency

Smart Insights (v5.42.0): - AI anomaly detection (usage spikes, unusual patterns) - Trend analysis with growth predictions - Achievement tracking (delight score peaks, retention milestones) - Actionable recommendations - Severity indicators (low/medium/high) - Confidence scores per insight

Brand Kit (v5.42.0): - Logo upload with drag-and-drop - Company name and tagline customization - Color pickers (Primary/Secondary/Accent) - Font selection for headers and body - Quick preset themes - Live preview card

Think Tank Example Prompts: - "Generate a user engagement report showing active users and session trends" - "Create a delight score analysis for the past month" - "Build a conversation analytics report with message volumes" - "Show me user retention metrics with churn analysis"

Usage: 1. Navigate to Reports → AI Writer tab 2. Select report style 3. Type or speak your request (click mic for voice) 4. Review generated report preview 5. Use modification prompt to refine 6. Toggle Edit Mode to modify sections 7. Export to PDF/Excel/HTML

Report Sections Generated: | Type | Description | |——|————-| | `heading` | H1-H3 headings | | `paragraph` | Body text | | `metrics` | 4-column KPI cards with trends (↑↓) | | `chart` | Interactive chart placeholders | | `table` | Data tables with headers | | `list` | Bullet point lists | | `quote` | Blockquote sections |

51.5 Table Categories

Category	Description
Core	Users, tenants, sessions
AI	Models, prompts, responses
Billing	Subscriptions, credits, invoices
Analytics	Events, metrics, logs
System	Configuration, audit, health

51.5 Field Options

Option	Description
Aggregation	none, count, sum, avg, min, max, distinct
Format	text, number, currency, percentage, date, datetime
Filter	Where clause conditions
Group By	Grouping columns

Option	Description
Order By	Sort columns and direction

51.6 Export Formats

Format	Description
CSV	Comma-separated values
JSON	Structured data format

51.7 API Endpoints

Base: `/api/admin/dynamic-reports`

Method	Endpoint	Description
`GET`	`/schema`	Discover database schema
`GET`	`/suggestions`	AI-generated report templates
`GET`	`/`	List saved reports
`POST`	`/`	Save report definition
`POST`	`/execute`	Execute a report
`POST`	`/export`	Export report data
`DELETE`	`/:id`	Delete a report

51.8 Implementation Files

File	Purpose
`apps/thinktank-admin/app/(dashboard)/reports/page.tsx`	Reports page
`packages/infrastructure/lambda/shared/services/schema-adaptive-reports.service.ts`	Backend service
`packages/infrastructure/lambda/admin/dynamic-reports.ts`	API handler
`packages/infrastructure/migrations/V2026_01_21_003__dynamic_reports.sql`	Database migration

Section 52: Gateway Status (v5.39.0)

Location: Think Tank Admin → Gateway

Monitor API Gateway health and traffic metrics.

52.1 Overview

The Gateway Status dashboard displays:

Metric	Description
Status	Overall gateway health (healthy/degraded/down)
Requests/sec	Current request throughput
Avg Latency	Mean response time
Error Rate	Percentage of failed requests
Active Connections	Current WebSocket connections

52.2 Endpoint Health

Column	Description
Endpoint	API route path
Method	HTTP method (GET, POST, etc.)
Status	healthy, slow, error
Latency	P50/P95/P99 response times
Requests	Request count (24h)
Errors	Error count (24h)

52.3 Traffic Patterns

View	Description
Hourly	Requests per hour (24h)
Daily	Requests per day (30d)
By Endpoint	Traffic distribution by route
By Status	Success vs error breakdown

52.4 Alerts

Alert Type	Trigger
High Latency	P95 > 2000ms
High Error Rate	Errors > 5%
Throughput Spike	2x normal traffic
Connection Drop	WebSocket disconnections

52.5 Implementation Files

File	Purpose
`apps/thinktank-admin/app/(dashboard)/gateway/page.tsx`	Gateway dashboard

Section 53: Decision Intelligence Artifacts (DIA Engine) (v5.43.0)

Location: Think Tank Admin → Decision Records

The Glass Box Decision Engine - transforms AI conversations into auditable, evidence-backed decision records with full provenance tracking.

53.1 Overview

Decision Intelligence Artifacts (DIA) solve the critical problem of AI decision opacity:

Challenge	DIA Solution
Black Box Decisions	Full claim-to-evidence mapping
Data Staleness	Volatile query tracking with automatic validation
Dissent Hidden	Ghost paths visualize rejected alternatives
Compliance Gaps	Built-in HIPAA/SOC2/GDPR export packages
Trust Uncertainty	Breathing heatmap shows trust topology at-a-glance

53.2 Core Concepts

Claims: Extracted conclusions, findings, recommendations, warnings, and facts from AI responses.

Claim Type	Description
`conclusion`	Final determination or decision
`finding`	Discovered information or observation
`recommendation`	Suggested course of action
`warning`	Risk or caution indicator
`fact`	Verified data point
`clinical_finding`	Healthcare-specific observation
`treatment_recommendation`	Medical treatment suggestion
`risk_assessment`	Risk evaluation
`legal_opinion`	Legal interpretation
`compliance_finding`	Regulatory compliance observation

Evidence Links: Connections between claims and their supporting data sources.

Evidence Type	Description
`tool_call`	API or tool execution result
`web_search`	Web search results
`document`	Referenced document
`calculation`	Computed result
`model_consensus`	Multiple model agreement

Dissent Events: Captured disagreements from model reasoning traces.

Severity	Description
`minor`	Small qualification or caveat
`moderate`	Significant alternative consideration
`significant`	Major disagreement requiring attention

Volatile Queries: Tool calls that may return different results over time.

Volatility	Threshold	Examples
`real-time`	1 hour	Stock prices, weather
`daily`	24 hours	News, analytics
`weekly`	168 hours	Document searches
`stable`	No expiry	Static references

53.3 The Living Parchment UI

The artifact viewer uses sensory design principles:

Breathing Heatmap Scrollbar: - Green (verified) - 6 BPM breathing rate - Amber (unverified) - Standard breathing - Red (contested) - 12 BPM alert breathing - Purple (stale) - Fading intensity with age

Living Ink Typography: - Font weight: 350-500 based on confidence (0-100%) - Stale claims fade to grayscale - Hover reveals evidence connections

Control Island (floating lens selector): - **Read**: Standard document view - **X-Ray**: Evidence links visible - **Risk**: Ghost paths and contested claims highlighted - **Compliance**: Regulatory framework coverage

Ghost Paths: Dashed connectors showing rejected alternatives from dissent events.

53.4 Artifact Lifecycle

```
Conversation → Extract → Validate → Active → [Validate] → Verified
                                       ↓
                                   Stale → Invalidated
                                     ↓
                                  Frozen (immutable)
```

Status	Description
`active`	Current, editable artifact
`frozen`	Immutable version with content hash
`archived`	Soft-deleted
`invalidated`	Data significantly changed

Validation Status	Description
`fresh`	Newly created, not yet validated
`stale`	Volatile queries exceeded thresholds
`verified`	Recently validated, data unchanged
`invalidated`	Significant data changes detected

53.5 Compliance Exports

Format	Use Case
`pdf`	Human-readable document
`json`	Machine-readable data
`hipaa_audit`	HIPAA compliance package with PHI inventory
`soc2_evidence`	SOC2 control mapping and evidence chain
`gdpr_dsar`	GDPR Data Subject Access Request response

HIPAA Audit Package Contents: - Cover sheet with artifact metadata - PHI inventory with categories - Access log for minimum necessary compliance - Evidence chain verification - System attestation with content hash

SOC2 Evidence Bundle Contents: - Control mapping (CC6.x, CC7.x, CC8.x) - Evidence chain completeness verification - Change management documentation - Integrity verification with signature

53.6 Configuration

Location: Think Tank Admin → Decision Records → Config

Setting	Default	Description
`diaEnabled`	`true`	Enable DIA Engine
`autoGenerateEnabled`	`false`	Auto-generate artifacts from conversations
`phiDetectionEnabled`	`true`	Scan for protected health information
`piiDetectionEnabled`	`true`	Scan for personally identifiable information
`defaultStalenessThresholdDays`	`1`	Days before volatile queries flagged stale
`maxArtifactsPerUser`	`0`	Limit per user (0 = unlimited)
`extractionModel`	`claude-3-5-sonnet`	Model for claim extraction
`autoRedactPhiOnExport`	`false`	Automatically redact PHI on export

53.7 Templates

Pre-configured extraction templates:

Template	Description	Compliance
General Decision Record	Standard extraction	None
Healthcare Decision	HIPAA-compliant clinical	HIPAA
Financial Analysis	Audit-ready financial	SOC2
Legal Review	Legal opinion documentation	SOC2, GDPR
Research Synthesis	Multi-source research	None

53.8 API Endpoints

Base: `/api/thinktank/decision-artifacts`

Method	Endpoint	Description
`GET`	`/`	List artifacts (supports filters)
`POST`	`/`	Generate artifact from conversation
`GET`	`/dashboard`	Dashboard metrics
`GET`	`/templates`	List available templates
`GET`	`/config`	Get tenant configuration
`PUT`	`/config`	Update configuration
`GET`	`/:id`	Get artifact details
`DELETE`	`/:id`	Archive artifact
`GET`	`/:id/staleness`	Check staleness status
`POST`	`/:id/validate`	Validate volatile queries
`POST`	`/:id/export`	Export artifact
`GET`	`/:id/versions`	Get version history
`GET`	`/:id/validation-history`	Validation audit trail
`GET`	`/:id/export-history`	Export audit trail

53.9 Dashboard Metrics

Metric	Description
Total Artifacts	All artifacts for tenant
Active Artifacts	Non-archived, non-frozen
Frozen Artifacts	Immutable versions
Average Confidence	Mean confidence across active
Stale Artifacts	Needing validation
PHI/PII Detected	Artifacts with sensitive data
Validation Cost MTD	API costs for re-validation
Top Domains	Most common primary domains
Compliance Usage	Framework distribution

53.10 Database Schema

Tables:

Table	Purpose
`decision_artifacts`	Main artifact storage
`decision_artifact_validation_log`	Validation audit trail
`decision_artifact_export_log`	Export audit trail
`decision_artifact_config`	Tenant configuration
`decision_artifact_templates`	Extraction templates
`decision_artifact_access_log`	Access audit (HIPAA)

Key Columns (`decision_artifacts`):

Column	Type	Description
`artifact_content`	JSONB	Claims, evidence, dissent, metrics
`heatmap_data`	JSONB	Pre-computed heatmap segments
`validation_status`	VARCHAR	fresh/stale/verified/invalidated
`phi_detected`	BOOLEAN	Contains protected health info
`content_hash`	VARCHAR(64)	SHA-256 for frozen artifacts

53.11 Implementation Files

File	Purpose
`packages/shared/src/types/decision-artifacts.ts`	Type definitions
`packages/infrastructure/lambda/shared/services/dia/`	Backend services
`packages/infrastructure/lambda/thinktank/decision-artifacts.ts`	API handler
`packages/infrastructure/lib/stacks/dia-stack.ts`	CDK infrastructure
`packages/infrastructure/migrations/V2026_01_22_001__decision_artifacts.sql`	Schema
`packages/infrastructure/migrations/V2026_01_22_002__decision_artifact_versioning.sql`	Versioning functions
`packages/infrastructure/migrations/V2026_01_22_003__decision_artifact_config.sql`	Config & templates
`apps/thinktank-admin/app/(dashboard)/decisions/`	Admin UI
`apps/thinktank-admin/app/(dashboard)/decisions/components/`	UI components

53.12 Troubleshooting

Issue	Solution
Extraction fails	Check Bedrock model access permissions
Missing evidence links	Verify tool_calls in message metadata
Stale status not updating	Run manual validation or check thresholds
PHI not detected	Verify phiDetectionEnabled in config
Export fails	Check S3 bucket permissions (DIA_EXPORT_BUCKET)
Version history empty	Artifact must be frozen to create versions

53.13 Security Considerations

- All tables have RLS policies enforcing tenant isolation

- PHI/PII detection runs automatically on extraction
- Access logging enabled for HIPAA compliance
- Content hashes provide tamper evidence for frozen artifacts
- Export audit trail tracks all compliance exports
- Presigned URLs expire after 1 hour

Section 54: Living Parchment 2029 Vision (v5.44.0)

Overview

Living Parchment is a comprehensive suite of advanced decision intelligence tools featuring sensory UI elements that communicate trust, confidence, and data freshness through visual breathing, living typography, and ghost paths. This 2029 Vision implementation transforms how users interact with AI-assisted decision making.

Design Philosophy

Concept	Implementation
Breathing Interfaces	UI elements pulse with life—faster breathing (12 BPM) indicates uncertainty, slower (4-6 BPM) indicates confidence
Living Ink	Text weight varies 350-500 based on confidence; stale information fades to grayscale
Ghost Paths	Rejected alternatives remain visible as translucent traces showing what could have been
Confidence Terrain	3D topographic visualization where elevation = confidence, color = risk

54.1 War Room (Strategic Decision Theater)

High-stakes collaborative decision space with AI advisors and confidence terrain.

Features

- **Confidence Terrain**: 3D grid visualization showing confidence topology across decision space
- **AI Advisory Council**: Multiple AI models providing different perspectives
- **Decision Paths**: Branching options with outcome predictions and advocate tracking
- **Ghost Branches**: Rejected paths remain visible for context
- **Stake Level Indicators**: Visual urgency based on decision importance

Advisor Types

Type	Color	Use Case
AI Model	Blue (#3b82f6)	Claude, GPT for strategic analysis
Human Expert	Purple (#8b5cf6)	Domain specialists
Domain Specialist	Cyan (#06b6d4)	Industry-specific advisors

Type	Color	Use Case

API Endpoints

```
POST   /api/thinktank/living-parchment/war-room           Create session
GET    /api/thinktank/living-parchment/war-room           List sessions
GET    /api/thinktank/living-parchment/war-room/:id       Get session
POST   /api/thinktank/living-parchment/war-room/:id/advisors        Add advisor
POST   /api/thinktank/living-parchment/war-room/:id/advisors/:aid/analyze   Request analysis
POST   /api/thinktank/living-parchment/war-room/:id/paths           Propose path
POST   /api/thinktank/living-parchment/war-room/:id/decide          Make decision
POST   /api/thinktank/living-parchment/war-room/:id/terrain         Update terrain
```

54.2 Council of Experts

Multi-persona AI consultation with consensus tracking and dissent visualization.

Expert Personas

Persona	Specialization	Style
Pragmatist	Practical Implementation	Results-focused, cost-conscious
Ethicist	Moral Philosophy	Principle-based, stakeholder-aware
Innovator	Creative Solutions	Visionary, possibility-focused
Skeptic	Risk Analysis	Devil's advocate, challenging
Synthesizer	Integration	Bridge-building, pattern-finding
Analyst	Data-Driven	Quantitative, evidence-based
Strategist	Long-term Strategy	Big-picture, competitive-aware
Humanist	Human Impact	Empathetic, user-centered

Consensus Visualization

- Experts positioned on circular visualization
- Positions move toward center as consensus increases
- Dissent sparks appear as electrical arcs between disagreeing experts
- Gravitational attraction animation shows convergence

API Endpoints

```
POST   /api/thinktank/living-parchment/council            Convene council
GET    /api/thinktank/living-parchment/council/:id        Get session
POST   /api/thinktank/living-parchment/council/:id/debate Run debate round
POST   /api/thinktank/living-parchment/council/:id/conclude Conclude session
```

54.3 Debate Arena

Adversarial exploration with attack/defense flows and steel-man generation.

Features

- **Resolution Meter**: Balance indicator (-100 to +100) showing which side is winning
- **Argument Flow**: Visual stream of claims, rebuttals, and concessions
- **Weak Point Detection**: Breathing red indicators on vulnerable arguments
- **Steel-Man Generation**: AI creates strongest version of opponent's argument
- **Attack/Defense Arrows**: Animated flows showing which arguments target which

Debate Phases

1. **Setup** - Configure debaters and proposition
2. **Opening** - Initial statements
3. **Main** - Core argument exchange
4. **Rebuttal** - Direct challenges
5. **Closing** - Final positions
6. **Resolved** - Outcome determined

API Endpoints

```
POST   /api/thinktank/living-parchment/debate             Create debate
GET    /api/thinktank/living-parchment/debate/:id         Get arena
POST   /api/thinktank/living-parchment/debate/:id/round   Run round
POST   /api/thinktank/living-parchment/debate/:id/steel-man Generate steel-man
```

54.4 Memory Palace (Coming Soon)

Navigable 3D knowledge topology with freshness fog.

- **Knowledge Rooms**: Domain-organized 3D spaces
- **Freshness Fog**: Stale areas appear foggy
- **Connection Threads**: Luminous lines between related concepts
- **Discovery Hotspots**: Breathing beacons where insights could emerge

54.5 Oracle View (Coming Soon)

Predictive confidence landscape.

- **Probability Heatmap**: Future timeline with brightness = confidence
- **Bifurcation Points**: Animated forks showing cascade effects
- **Ghost Futures**: Translucent overlays of alternative scenarios
- **Black Swan Indicators**: Dormant embers for low-probability/high-impact events

54.6 Synthesis Engine (Coming Soon)

Multi-source fusion view.

- **Source Streams**: Flowing rivers converging into synthesis
- **Agreement Zones**: Warm glow where sources align
- **Tension Zones**: Crackling energy between contradictions
- **Provenance Trails**: Click any claim to see all supporting sources

54.7 Cognitive Load Monitor (Coming Soon)

User state awareness with adaptive UI.

- **Attention Heatmap**: Track where user has focused
- **Fatigue Indicators**: UI breathing slows as session lengthens
- **Overwhelm Warning**: Screen edges breathe red when load peaks

54.8 Temporal Drift Observatory (Coming Soon)

Fact evolution tracking.

- **Drift Alerts**: Notifications when facts have changed
- **Version Ghosts**: Previous versions as translucent overlays
- **Citation Half-Life**: Predict when facts likely become stale

Database Schema

```
-- Core tables (see migration V2026_01_22_004)
war_room_sessions, war_room_participants, war_room_advisors
memory_palaces, memory_rooms, knowledge_nodes, memory_connections
oracle_views, oracle_predictions, bifurcation_points, ghost_futures
synthesis_sessions, synthesis_sources, synthesis_claims
cognitive_load_sessions, cognitive_load_history
council_sessions, council_experts, expert_arguments, minority_reports
drifting_facts, drift_alerts, version_ghosts
debate_arenas, debaters, debate_arguments, weak_points, steel_man_overlays
living_parchment_config
```

Configuration

```
interface LivingParchmentConfig {
  features: {
    warRoomEnabled: boolean;          // Default: true
    memoryPalaceEnabled: boolean;     // Default: true
    oracleViewEnabled: boolean;       // Default: true
    synthesisEngineEnabled: boolean;  // Default: true
    cognitiveLoadEnabled: boolean;    // Default: true
    councilOfExpertsEnabled: boolean; // Default: true
    temporalDriftEnabled: boolean;    // Default: true
    debateArenaEnabled: boolean;      // Default: true
  };
```

```
  defaults: {
    breathingRateBase: 6;              // BPM
    confidenceThreshold: 70;           // Minimum for "high confidence"
    stalenessThresholdDays: 30;        // When facts become stale
    maxAdvisors: 10;
    maxExperts: 8;
    maxDebateRounds: 5;
  };
  visualSettings: {
    heatmapColorScheme: 'standard' | 'accessible' | 'dark';
    animationIntensity: 'subtle' | 'normal' | 'vivid';
    ghostOpacity: 0.5;
  };
}
```

Implementation Files

```
packages/shared/src/types/living-parchment.types.ts     # All types
packages/infrastructure/migrations/V2026_01_22_004__living_parchment_core.sql
packages/infrastructure/lambda/shared/services/living-parchment/
    war-room.service.ts
    council-of-experts.service.ts
    debate-arena.service.ts
    index.ts
packages/infrastructure/lambda/thinktank/living-parchment.ts
apps/thinktank-admin/app/(dashboard)/living-parchment/
    page.tsx                          # Landing page
    war-room/page.tsx                 # War Room UI
    council/page.tsx                  # Council of Experts UI
    debate/page.tsx                   # Debate Arena UI
```

Security Considerations

- All tables have RLS policies enforcing tenant isolation
- AI advisor calls use tenant-scoped model access
- Session ownership validated before modifications
- Audit logging for all decision actions
- Debate content filtered for appropriate use

---

45. Localization & Translation Overrides

Location: Think Tank Admin → Administration → Localization

Tenant administrators can customize UI text and messages across Think Tank with translation overrides.

45.1 Overview

The localization system allows tenants to: - Override any system string with custom text - Protect overrides from automatic translation updates - Configure default and enabled languages for users - Maintain brand consistency across all 18 supported languages

45.2 Supported Languages

Language	Code	Flag
English	en	
Spanish	es	
French	fr	
German	de	
Portuguese	pt	
Italian	it	
Dutch	nl	
Polish	pl	
Russian	ru	
Turkish	tr	
Japanese	ja	
Korean	ko	
Chinese (Simplified)	zh-CN	
Chinese (Traditional)	zh-TW	
Arabic	ar	
Hindi	hi	
Thai	th	
Vietnamese	vi	

45.3 Admin UI Tabs

Tab	Purpose
Your Overrides	View and manage custom translations
Browse Strings	Search Think Tank strings to customize
Configuration	Set default language and enabled languages

45.4 Creating Translation Overrides

1. Navigate to **Administration → Localization**
2. Select target language from dropdown
3. Go to **Browse Strings** tab
4. Search for the string you want to customize
5. Click **Edit** to open the override dialog
6. Enter your custom text
7. Toggle **Protect from automatic updates** (recommended)
8. Click **Save**

45.5 Protection System

Protected overrides (default): - Will NOT be updated when system translations improve - Recommended for brand-specific terminology - Shows lock icon in override list

Unprotected overrides: - May be updated by automatic translation systems - Useful for temporary fixes until system improves - Shows unlock icon in override list

Reverting to system translation: - Click the revert button on any override - Override is deleted and system translation is restored

45.6 Language Configuration

Configure which languages are available to your users:

1. Go to **Configuration** tab
2. Set **Default Language** for new users
3. Enable/disable languages by clicking language cards
4. At least one language must remain enabled

45.7 Common Use Cases

Use Case	Example
Brand terminology	Replace "Think Tank" with your product name
Industry jargon	Use domain-specific terms
Tone adjustment	Make messages more formal/casual
Legal compliance	Customize disclaimers
Regional variants	UK vs US English differences

45.8 API Reference

Base URL: `/api/admin/localization`

Endpoint	Method	Purpose
`/overrides`	GET	List your overrides
`/overrides`	POST	Create/update override
`/overrides/:id`	DELETE	Revert to system
`/overrides/:id/protection`	PATCH	Toggle protection
`/config`	GET/PUT	Language configuration
`/bundle/:lang`	GET	Get translations with overrides

45.9 Database Tables

Table	Purpose
`tenant_translation_overrides`	Custom translations per tenant
`tenant_localization_config`	Language settings per tenant
`translation_audit_log`	Change history

45.10 Implementation Files

```
packages/infrastructure/migrations/V2026_01_25_006__tenant_translation_overrides.sql
packages/infrastructure/lambda/admin/localization-registry.ts
apps/thinktank-admin/app/(dashboard)/localization/page.tsx
```

Related Documentation

- RADIANT Admin Guide - Platform administration
- RADIANT Admin Guide - HITL Orchestration - Full HITL Orchestration documentation
- RADIANT Admin Guide - Metrics & Learning - Persistent learning system
- RADIANT Admin Guide - Consciousness Evolution - Predictive coding, LoRA evolution, Local Ego
- Think Tank User Guide - End user guide
- User Rules System - Memory rules details
- Provider Rejection Handling - Rejection system
- AI Ethics Standards - Ethics framework