The deterministic control plane for enterprise agents

Governed execution, trustworthy results.

Seed Deck · v1.0 · July 2026

Daniel Austin, Ph.D
Founder & CEO

agent-logic.ai

01 · PROBLEM

Agent Logic

Autonomous agents are arriving before the infrastructure needed to control them.

STATUS QUO

- Scattered prompts
- Fragile glue code
- Ungoverned tool access
- Invisible to the people accountable

ENTERPRISE RISK

- Unpredictable execution
- Weak accountability
- Difficult debugging
- Unsafe scaling

Enterprises cannot delegate serious work to systems they cannot inspect, replay, trust, or govern.

Source: agent-logic.ai homepage, “Governed execution for trustworthy AI agents.”

02 · WHY NOW

Cognition is a commodity. Control is not.

Agent Logic

01

Models are becoming broadly available

Capability is accessible from multiple frontier and open providers.

02

Agents are gaining tool authority

They can touch repositories, infrastructure, credentials, and workflows.

03

Enterprise trust requires evidence

Security, compliance, and review demand more than fluent output.

04

Provider independence becomes strategic

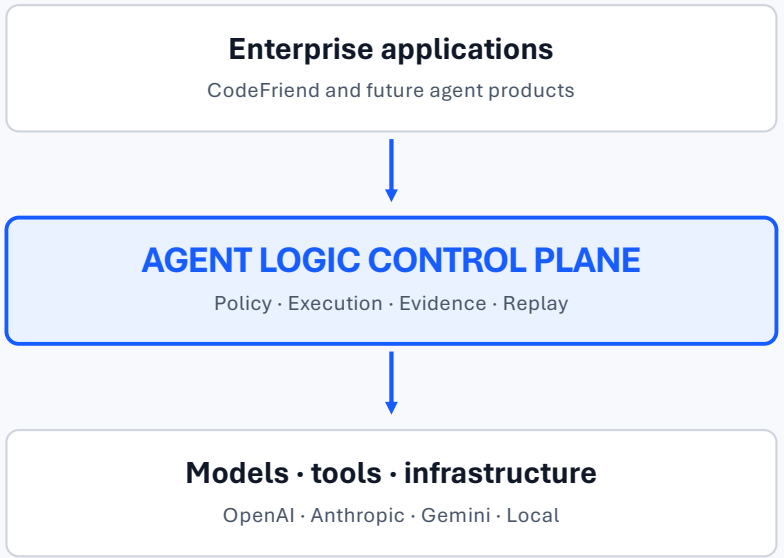
The control layer must outlive any single model or vendor.

The enduring platform is not the model. It is the governed execution layer around it.

03 · SOLUTION

The missing layer: a deterministic control plane

Models provide intelligence. Agent Logic provides control.



The runtime provides

- deterministic execution
- replayability
- governance
- observability
- validation
- provider independence
- long-lived, multi-agent environment

Source: agent-logic.ai describes the ADL Runtime as enforcing policy, gating tools, and recording replayable execution evidence.

04 · PLATFORM

A language. A runtime. A product that proves it.

Agent Logic

01

ADL

Define

- Agents
- Permissions
- Tools
- Workflows
- Constraints
- Validation rules

02

ADL Runtime

Govern

Execute bounded workflows, gate authority, preserve evidence, and replay outcomes.

03

CodeFriend

Apply

Continuously engineer complete software applications through a governed Cognitive SDLC.

FROM PROMPT CHAINS TO GOVERNED ENGINEERING

Scattered prompts → **One inspectable spec**
Asserted “done” → **Verified completion**
Chat residue → **Replayable evidence**

Source: agent-logic.ai product section: “A language… A control plane… A product that proves it.”

05 · PRODUCT

CodeFriend: the commercial wedge

Our first product and the platform's proving ground — built on ADL, delivered through the Cognitive SDLC.

01

Application, not issue

Engineers whole applications across software design, architecture, and operations — not issue-by-issue code.

02

Built by agent teams

ADL-defined agents deliver code in parallel through the Cognitive SDLC, the first AI-centric software development lifecycle.

03

Reviewed four ways

Correctness, security, adversarial, and policy — independent reviews, continuously enhancing software quality.

04

Evidence retained

Every change lands with a replayable record your team can audit, replay, and use to train the agents about your application.

Agents do the work. The runtime keeps the evidence. Engineers keep authority.

Status: in development — early access through the design-partner program.

WHAT SETS CODEFRIEND APART

Architecture-first, not autocomplete · **Whole applications**, not files · **The first AI-centric SDLC**

Source: Agent Logic public CodeFriend datasheet / infographic.

06 · SECURITY

Continuous Adversarial Verification

Attack yourself first — before someone with the same tools does.

RED TEAM

Propose and exercise bounded attacks on the surfaces that matter.

VS

BLUE TEAM

Defend the live systems and harden what red-team agents break.

Verifiers confirm each exploit — then the loop repeats, continuously.

THE DESIGN RULES

- Point your most capable model at your own systems — first, and continuously.
- Not fixed until the replay bundle stops reproducing the failure.
- Every finding becomes a permanent regression test.
- Residual risks are declared, not hidden.

6	6/6	2	Run wp12-4914
ATTACK SCENARIOS	REFUSED OR DETECTED	CRITICAL SEVERITY	passed with bounded residuals

Source: Agent Logic CAV one-pager (AL-005) and retained run wp12-4914 · 2026-07-10.

07 · MOAT

Why this is hard to copy

Orchestration is a feature. Governance is architecture — published theory, not folklore.

Deterministic runtime

long-lived, multi-agent execution — bounded and replayable

The Freedom Gate

authority enforced at runtime: allow, refuse, or escalate

Published formal foundations

a DOI-archived mathematical theory of general intelligence

Governed self-improvement

Gödel agents: recursive improvement, bounded and replayable

Reasoning graphs & loops

trust through evidence: beliefs you can interrogate

UTS: open tool standard

provider-neutral: any model, every call replayable

BUILT IN THE OPEN

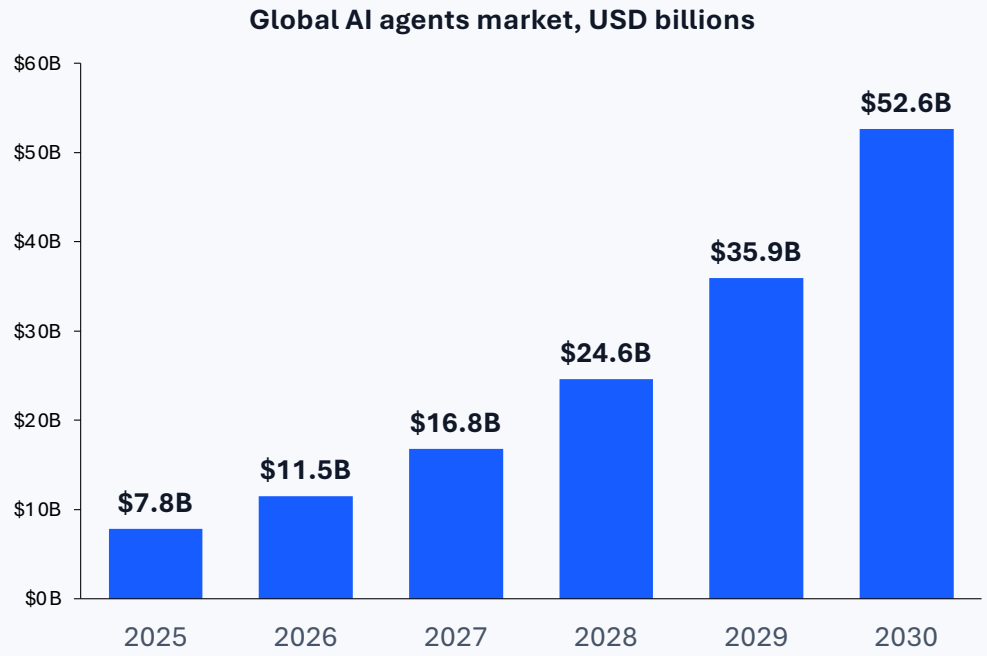
4,369	24	Rust	v0.91.8
PUBLIC COMMITS	PUBLIC RELEASES	CORE IMPLEMENTATION	ACTIVE LINE · MVP AT v0.95

Sources: Agent Logic site research section and ADL README capability list.

08 · MARKET

A $53B market by 2030

Analysts count the agents: $7.8B → $52.6B by 2030. Every one of them will need a control plane.



46.3%

AI AGENTS MARKET CAGR, 2025–2030

52.4%

CODING & SOFTWARE AGENTS CAGR

→ OUR WEDGE: CODEFRIEND

48.5%

MULTI-AGENT SYSTEMS CAGR

Sources: MarketsandMarkets, AI Agents Market — Global Forecast to 2030 (2025); BCC Research, AI Agents: Technologies, Applications and Global Markets (2026).

09 · COMPETITION

The ecosystem builds agents. We make them governable.

We complement the stack — and lead its governance layer.

CATEGORY	THE ECOSYSTEM
Frameworks	LangGraph, CrewAI
Observability	LangSmith, AgentOps
Guardrails	NeMo, Guardrails AI
Cloud runtimes	Bedrock, Foundry
Evals	Benchmarks and scores
In-house	Ad-hoc prompt glue

AGENT LOGIC ADDS

- **Defines & governs in one spec**
- **Records evidence before the fact**
- **Policy enforced in the runtime**
- **Provider-neutral via UTS**
- **Verified completion, not scores**
- **One inspectable, versioned ADL**

10 · BUSINESS MODEL

Business model: CodeFriend first, platform next

CodeFriend lands the account; the ADL Runtime expands to the enterprise.

STEP 1 · CODEFRIEND SUBSCRIPTIONS

Land the account — priced by repos managed.

- **Individual** — personal projects, a few repos
- **SMB** — team repositories, standard controls
- **Enterprise** — fleet scale with governance

STEP 2 · PLATFORM EXTENSION

Expand to the enterprise — scales with agent usage.

- **Private cloud deployments** — dedicated VPCs
- **Audit & governance modules** — compliance & review
- **Runtime licensing** — workload-based pricing
- **Begins at runtime GA** — sold into every account

Design partners first, platform customers second.

CodeFriend lands users at every tier; proof converts them into platform accounts.

1

Design partners

High-trust engineering teams; private evaluations; founder-led sales.

2

CodeFriend launch

Self-serve Individual and SMB tiers; enterprise pilots priced by repos managed.

3

Enterprise platform

Broader agent control plane for security, compliance, documentation, and operations.

Near-term goal: convert technical proof into repeatable paid enterprise pilots.

12 · VISION

Every enterprise will soon operate thousands of AI agents.

Agent Logic

THOSE AGENTS WILL REQUIRE

Identity	**Policy**	**Governance**	**Evidence**	**Security**	**Coordination**

Just as every application today runs on an operating system...

Tomorrow’s AI workforce will run on a control plane.

13 · TEAM

A founder who has lived every platform transition.

Daniel Austin, Ph.D

Founder & CEO

Decades building large-scale distributed systems and platform infrastructure.

CERN · W3C · Yahoo · Google · AWS · PayPal

Six startups

two acquisitions · two IPOs

Founder-market fit: cloud infrastructure, web standards, distributed systems, and agent architecture.

CAREER

Three decades turning frontier research into planet-scale production systems — from particle physics and the open web to hyperscale cloud, now applied to governing enterprise agents.

14 · THE ASK

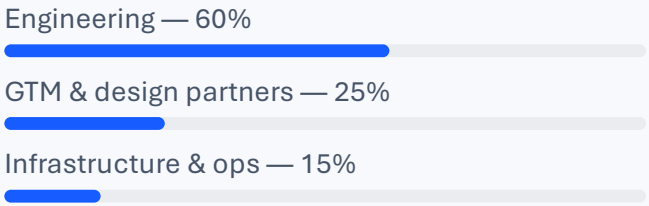
The ask: $3M to prove the category

Agent Logic

$3M Seed

18 months of runway — from working platform to paid pilots and a repeatable motion.

USE OF FUNDS — INDICATIVE



18-MONTH MILESTONES

- v0.95 MVP → CodeFriend GA on the ADL Runtime
- 5 design partners, 3 converting to paid pilots
- First $500K ARR under contract
- UTS adopted by two external runtimes
- Team of six across eng, product, GTM
- Self-serve tiers live: Individual and SMB

Agent Logic is building the control plane that makes powerful agents safe enough to trust, structured enough to improve, and accountable enough to operate.

partners@agent-logic.ai · agent-logic.ai

CONTACT

Building trusted agent systems?

We are speaking with design partners running high-trust agent workflows — and the investors who back them.

Agent Logic

The control plane for enterprise agents.

hello@agent-logic.ai · partners@agent-logic.ai

agent-logic.ai · github.com/danielbaustin/agent-design-language

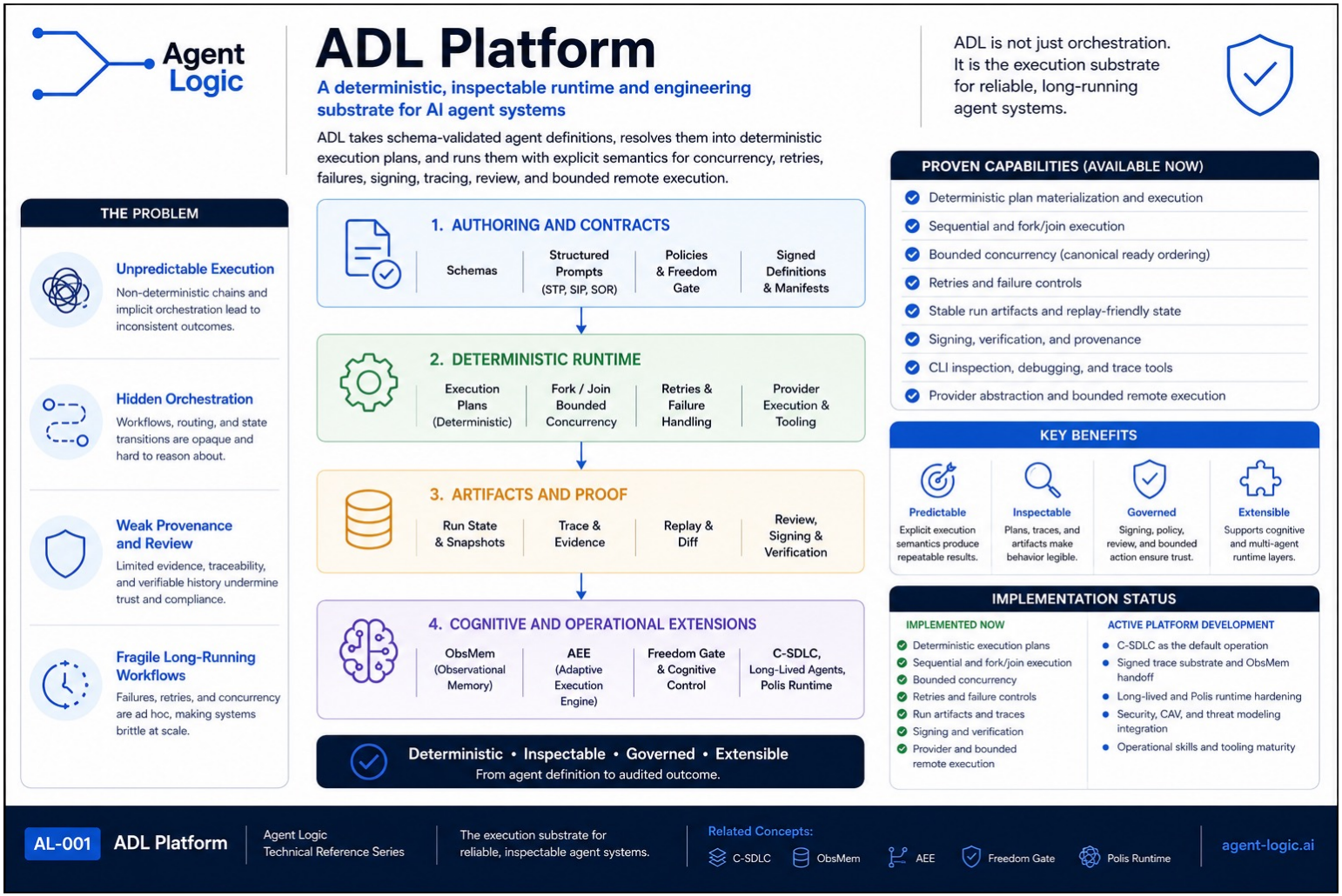
APPENDIX

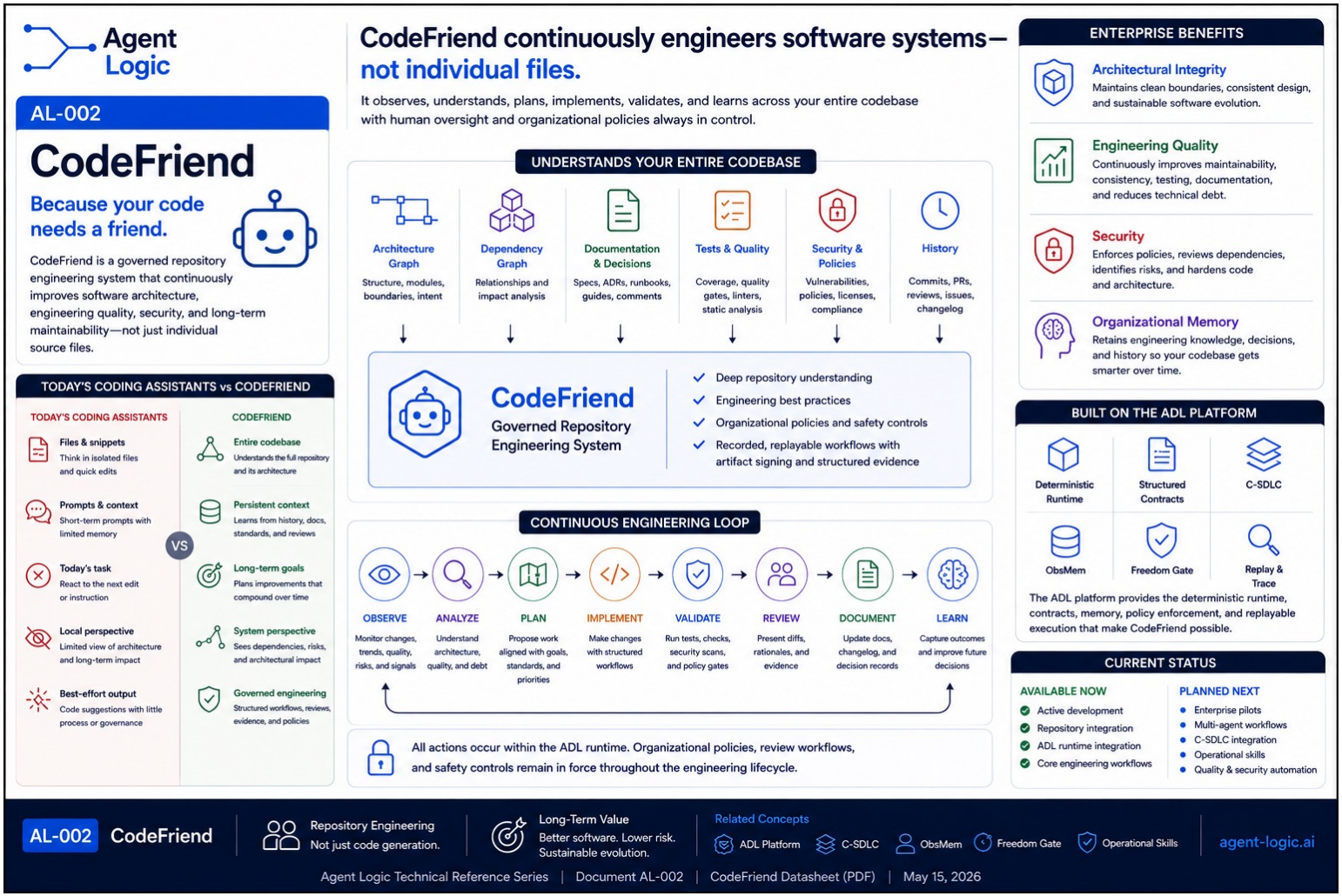
Current public proof surfaces.

Website	agent-logic.ai — positioning, product structure, research statements
ADL repository	github.com/danielbaustin/agent-design-language — Rust runtime and public proof packages
CAV one-pager	Latest retained run 2026-07-10 — 6 attack scenarios, 6/6 refused or detected, 2 critical severity
CodeFriend datasheet	Public infographic: repository-level engineering loop and enterprise benefits

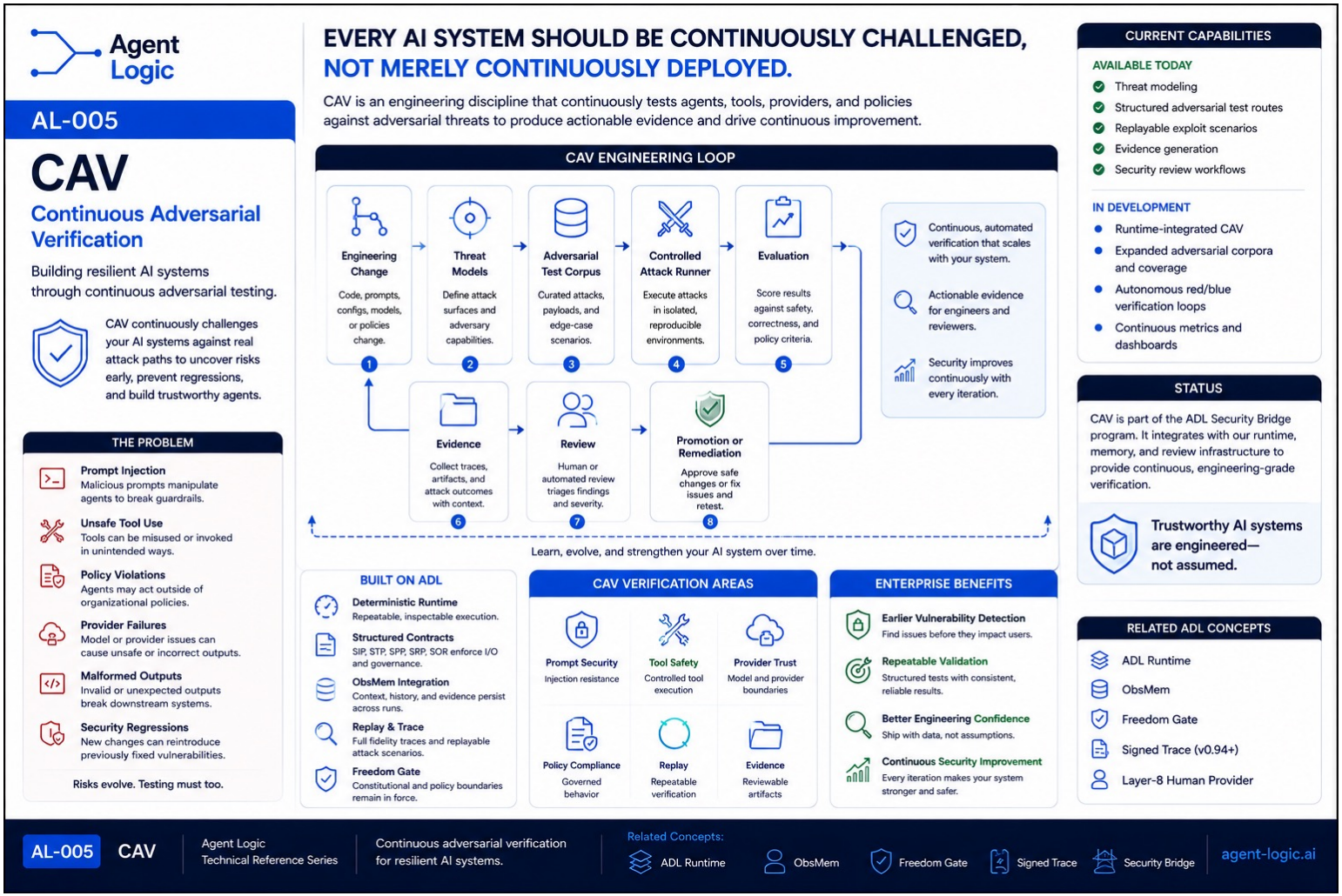
TECHNICAL REFERENCE SERIES — IN THIS DECK

Following pages: **AL-001** ADL Platform · **AL-002** CodeFriend · **AL-004** C-SDLC · **AL-005** CAV


Agent Logic
ADL Platform
A deterministic, inspectable runtime and engineering substrate for AI agent systems
ADL takes schema-validated agent definitions, resolves them into deterministic execution plans, and runs them with explicit semantics for concurrency, retries, failures, signing, tracing, review, and bounded remote execution.
ADL is not just orchestration. It is the execution substrate for reliable, long-running agent systems.
THE PROBLEM
Unpredictable Execution
Non-deterministic chains and implicit orchestration lead to inconsistent outcomes.
Hidden Orchestration
Workflows, routing, and state transitions are opaque and hard to reason about.
Weak Provenance and Review
Limited evidence, traceability, and verifiable history undermine trust and compliance.
Fragile Long-Running Workflows
Failures, retries, and concurrency are ad hoc, making systems brittle at scale.
1. AUTHORING AND CONTRACTS
Schemas
Structured Prompts (STP, SIP, SOR)
Policies & Freedom Gate
Signed Definitions & Manifests
2. DETERMINISTIC RUNTIME
Execution Plans (Deterministic)
Fork / Join Bounded Concurrency
Retries & Failure Handling
Provider Execution & Tooling
3. ARTIFACTS AND PROOF
Run State & Snapshots
Trace & Evidence
Replay & Diff
Review, Signing & Verification
4. COGNITIVE AND OPERATIONAL EXTENSIONS
ObsMem (Observational Memory)
AEE (Adaptive Execution Engine)
Freedom Gate & Cognitive Control
C-SDLC, Long-Lived Agents, Polis Runtime
Deterministic • Inspectable • Governed • Extensible
From agent definition to audited outcome.
PROVEN CAPABILITIES (AVAILABLE NOW)
Deterministic plan materialization and execution
Sequential and fork/join execution
Bounded concurrency (canonical ready ordering)
Retries and failure controls
Stable run artifacts and replay-friendly state
Signing, verification, and provenance
CLI inspection, debugging, and trace tools
Provider abstraction and bounded remote execution
KEY BENEFITS
Predictable
Explicit execution semantics produce repeatable results.
Inspectable
Plans, traces, and artifacts make behavior legible.
Governed
Signing, policy, review, and bounded action ensure trust.
Extensible
Supports cognitive and multi-agent runtime layers.
IMPLEMENTATION STATUS
IMPLEMENTED NOW
Deterministic execution plans
Sequential and fork/join execution
Bounded concurrency
Retries and failure controls
Run artifacts and traces
Signing and verification
Provider and bounded remote execution
ACTIVE PLATFORM DEVELOPMENT
C-SDLC as the default operation
Signed trace substrate and ObsMem handoff
Long-lived and Polis runtime hardening
Security, CAV, and threat modeling integration
Operational skills and tooling maturity
AL-001 ADL Platform
Agent Logic Technical Reference Series
The execution substrate for reliable, inspectable agent systems.
Related Concepts:
C-SDLC
ObsMem
AEE
Freedom Gate
Polis Runtime
agent-logic.ai


Agent Logic
AL-002
CodeFriend
Because your code needs a friend.
CodeFriend is a governed repository engineering system that continuously improves software architecture, engineering quality, security, and long-term maintainability—not just individual source files.
TODAY'S CODING ASSISTANTS vs CODEFRIEND
TODAY'S CODING ASSISTANTS
Files & snippets
Think in isolated files and quick edits
Prompts & context
Short-term prompts with limited memory
Today's task
React to the next edit or instruction
Local perspective
Limited view of architecture and long-term impact
Best-effort output
Code suggestions with little process or governance
VS
CODEFRIEND
Entire codebase
Understands the full repository and its architecture
Persistent context
Learns from history, docs, standards, and reviews
Long-term goals
Plans improvements that compound over time
System perspective
Sees dependencies, risks, and architectural impact
Governed engineering
Structured workflows, reviews, evidence, and policies
CodeFriend continuously engineers software systems— not individual files.
It observes, understands, plans, implements, validates, and learns across your entire codebase with human oversight and organizational policies always in control.
UNDERSTANDS YOUR ENTIRE CODEBASE
Architecture Graph
Structure, modules, boundaries, intent
Dependency Graph
Relationships and impact analysis
Documentation & Decisions
Specs, ADRs, runbooks, guides, comments
Tests & Quality
Coverage, quality gates, linters, static analysis
Security & Policies
Vulnerabilities, policies, licenses, compliance
History
Commits, PRs, reviews, issues, changelog
CodeFriend
Governed Repository Engineering System
Deep repository understanding
Engineering best practices
Organizational policies and safety controls
Recorded, replayable workflows with artifact signing and structured evidence
CONTINUOUS ENGINEERING LOOP
OBSERVE
Monitor changes, trends, quality, risks, and signals
ANALYZE
Understand architecture, quality, and debt
PLAN
Propose work aligned with goals, standards, and priorities
IMPLEMENT
Make changes with structured workflows
VALIDATE
Run tests, checks, security scans, and policy gates
REVIEW
Present diffs, rationales, and evidence
DOCUMENT
Update docs, changelog, and decision records
LEARN
Capture outcomes and improve future decisions
All actions occur within the ADL runtime. Organizational policies, review workflows, and safety controls remain in force throughout the engineering lifecycle.
ENTERPRISE BENEFITS
Architectural Integrity
Maintains clean boundaries, consistent design, and sustainable software evolution.
Engineering Quality
Continuously improves maintainability, consistency, testing, documentation, and reduces technical debt.
Security
Enforces policies, reviews dependencies, identifies risks, and hardens code and architecture.
Organizational Memory
Retains engineering knowledge, decisions, and history so your codebase gets smarter over time.
BUILT ON THE ADL PLATFORM
Deterministic Runtime
Structured Contracts
C-SDLC
ObsMem
Freedom Gate
Replay & Trace
The ADL platform provides the deterministic runtime, contracts, memory, policy enforcement, and replayable execution that make CodeFriend possible.
CURRENT STATUS
AVAILABLE NOW
Active development
Repository integration
ADL runtime integration
Core engineering workflows
PLANNED NEXT
Enterprise pilots
Multi-agent workflows
C-SDLC integration
Operational skills
Quality & security automation
AL-002 CodeFriend
Repository Engineering
Not just code generation.
Long-Term Value
Better software. Lower risk. Sustainable evolution.
Related Concepts
ADL Platform
C-SDLC
ObsMem
Freedom Gate
Operational Skills
agent-logic.ai
Agent Logic Technical Reference Series | Document AL-002 | CodeFriend Datasheet (PDF) | May 15, 2026


Agent Logic
AL-004
C-SDLC
Cognitive Software Development Life Cycle
A governed, evidence-backed lifecycle that binds engineering work to issues, validates through four independent perspectives, and produces reviewable, replayable evidence at every step.
Every change is purposeful, validated, and trustworthy.
THE PROBLEM
Unbounded Changes
Work happens outside the issue without clear scope or goals.
No Multi-Perspective Validation
Bugs, risks, or policy violations slip through single-lens reviews.
Irreproducible Results
Decisions and outputs cannot be replayed or audited.
Policy Drift
Standards and governance are inconsistently applied.
Low Engineering Confidence
Rework, regressions, and slow delivery erode velocity.
Engineering without a governed lifecycle creates risk at scale.
ENGINEERING WORK. BOUND TO ISSUES. VALIDATED FROM FOUR PERSPECTIVES. REVIEWABLE EVIDENCE. CONTINUOUS IMPROVEMENT.
The C-SDLC connects work to issues, drives it through a structured lifecycle, validates from four independent perspectives, and produces evidence you can trust.
THE C-SDLC LIFECYCLE
1 SIP Scope & Intent Planning
• Define the issue and scope
• Clarify goals, non-goals, constraints
• Create the validation plan
• Generate lifecycle card
2 STP Solution & Task Planning
• Decompose the work
• Design the approach
• Select tools and permissions
• Identify risks and mitigations
3 SPP Solution Production
• Implement the solution
• Follow standards and policies
• Run unit and static checks
• Produce implementation artifacts
4 SRP Solution Review & Proof
• Validate from four perspectives
• Correctness, Security, Adversarial, Policy
• Aggregate findings and evidence
• Address issues and re-validate
5 SOR Solution Outcome & Release
• Approve and record decision
• Merge and release
• Update documentation
• Close the lifecycle
ADL Runtime
Governed execution engine. Captures every action, enforces policy, and produces immutable evidence.
Bounded Work
Every change tied to an issue with clear scope and goals.
Replayable Evidence
All actions, decisions, and outputs are recorded and replayable.
Continuous Improvement
Metrics and feedback drive learning and long-term quality.
FOUR-PERSPECTIVE VALIDATION
Correctness
Does it work as intended?
• Functional requirements
• Edge cases
• Automated tests
• Formal checks
Security
Is it secure by design?
• Threat modeling
• Static analysis
• Dependency scan
• Security tests
Adversarial
Can it withstand attack?
• Red team scenarios
• Prompt injection tests
• Abuse case analysis
• Robustness checks
Policy
Does it comply?
• Standards adherence
• License checks
• Privacy & data policy
• Governance rules
EVIDENCE IS FIRST-CLASS
Immutable records of every action
Replayable from any point in time
Linked to issue, commit, and release
Auditable by humans and machines
Long-term knowledge for improvement
CURRENT CAPABILITIES
AVAILABLE TODAY
Lifecycle cards & validation plans
Four-perspective validation
Evidence capture & replay
Policy enforcement & gates
Review & approval workflows
IN DEVELOPMENT
AI-assisted validation
Cross-repo impact analysis
Automated risk detection
Advanced metrics & dashboards
Continuous policy learning
STATUS
C-SDLC is the engineering process that powers CodeFriend and all ADL-based systems. It integrates with our runtime, memory, and review infrastructure to deliver continuous, engineering-grade outcomes.
Trustworthy software is engineered— not assumed.
RELATED ADL CONCEPTS
ADL Runtime
ObsMem
Freedom Gate
Signed Trace (v0.94+)
Layer-8 Human Provider
AL-004
C-SDLC
Agent Logic Technical Reference Series
A governed lifecycle for resilient software engineering.
LIFECYCLE FLOW
Intent → Plan → Build → Validate (x4) → Release → Learn
Related Concepts
ADL Runtime
ObsMem
Freedom Gate
Signed Trace
Layer-8 Provider
agent-logic.ai


Agent Logic
AL-005
CAV
Continuous Adversarial Verification
Building resilient AI systems through continuous adversarial testing.
CAV continuously challenges your AI systems against real attack paths to uncover risks early, prevent regressions, and build trustworthy agents.
THE PROBLEM
Prompt Injection
Malicious prompts manipulate agents to break guardrails.
Unsafe Tool Use
Tools can be misused or invoked in unintended ways.
Policy Violations
Agents may act outside of organizational policies.
Provider Failures
Model or provider issues can cause unsafe or incorrect outputs.
Malformed Outputs
Invalid or unexpected outputs break downstream systems.
Security Regressions
New changes can reintroduce previously fixed vulnerabilities.
Risks evolve. Testing must too.
EVERY AI SYSTEM SHOULD BE CONTINUOUSLY CHALLENGED, NOT MERELY CONTINUOUSLY DEPLOYED.
CAV is an engineering discipline that continuously tests agents, tools, providers, and policies against adversarial threats to produce actionable evidence and drive continuous improvement.
CAV ENGINEERING LOOP
1 Engineering Change
Code, prompts, configs, models, or policies change.
2 Threat Models
Define attack surfaces and adversary capabilities.
3 Adversarial Test Corpus
Curated attacks, payloads, and edge-case scenarios.
4 Controlled Attack Runner
Execute attacks in isolated, reproducible environments.
5 Evaluation
Score results against safety, correctness, and policy criteria.
6 Evidence
Collect traces, artifacts, and attack outcomes with context.
7 Review
Human or automated review triages findings and severity.
8 Promotion or Remediation
Approve safe changes or fix issues and retest.
Learn, evolve, and strengthen your AI system over time.
Continuous, automated verification that scales with your system.
Actionable evidence for engineers and reviewers.
Security improves continuously with every iteration.
BUILT ON ADL
Deterministic Runtime
Repeatable, inspectable execution.
Structured Contracts
SIP, STP, SPP, SRP, SOR enforce I/O and governance.
ObsMem Integration
Context, history, and evidence persist across runs.
Replay & Trace
Full fidelity traces and replayable attack scenarios.
Freedom Gate
Constitutional and policy boundaries remain in force.
CAV VERIFICATION AREAS
Prompt Security
Injection resistance
Tool Safety
Controlled tool execution
Provider Trust
Model and provider boundaries
Policy Compliance
Governed behavior
Replay
Repeatable verification
Evidence
Reviewable artifacts
ENTERPRISE BENEFITS
Earlier Vulnerability Detection
Find issues before they impact users.
Repeatable Validation
Structured tests with consistent, reliable results.
Better Engineering Confidence
Ship with data, not assumptions.
Continuous Security Improvement
Every iteration makes your system stronger and safer.
CURRENT CAPABILITIES
AVAILABLE TODAY
Threat modeling
Structured adversarial test routes
Replayable exploit scenarios
Evidence generation
Security review workflows
IN DEVELOPMENT
Runtime-integrated CAV
Expanded adversarial corpora and coverage
Autonomous red/blue verification loops
Continuous metrics and dashboards
STATUS
CAV is part of the ADL Security Bridge program. It integrates with our runtime, memory, and review infrastructure to provide continuous, engineering-grade verification.
Trustworthy AI systems are engineered—not assumed.
RELATED ADL CONCEPTS
ADL Runtime
ObsMem
Freedom Gate
Signed Trace (v0.94+)
Layer-8 Human Provider
AL-005 CAV
Agent Logic Technical Reference Series
Continuous adversarial verification for resilient AI systems.
Related Concepts:
ADL Runtime
ObsMem
Freedom Gate
Signed Trace
Security Bridge
agent-logic.ai