TECHNICAL WHITEPAPER WP-07 · ENTERPRISE ARCHITECTURE, ENGINEERING TRADE-OFFS & BUYER VERIFICATION GUIDE

Enterprise Architecture, Engineering Trade-Offs & Buyer Verification Guide: Compiler Internals, Cached MCP Economics & Hands-On Proof

An unvarnished systems engineering guide for CTOs, CISOs, and Principal Architects-covering Arcverin's two-plane compiler and native Rust data plane, Curated Eager vs. Progressive 1-2-3 MCP under 90% prompt caching, distributed-systems realities, PostgreSQL 15-18 FORCE RLS, and a 6-step offline binary verification checklist.

2 Planes
AOT COMPILER + RUST RUNTIME
Offline WorkerIr compilation + self-hosted native Rust data plane.

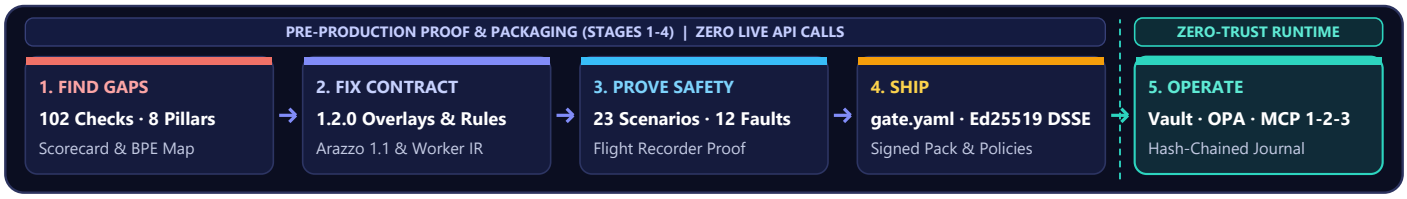
Eager First
PROMPT-CACHE MCP HONESTY
Curated --mode eager beats 1-2-3 on bounded domains & native Tool Search.

15 RLS Tables
POSTGRESQL 15-18 ISOLATION
FORCE ROW LEVEL SECURITY + SET LOCAL PgBouncer transaction safety.

6-Step Audit
30-MINUTE BINARY PROOF
Verify SHA-256, 102 rules, 12 faults, OPA fail-closed & journals offline.

1. Two-Plane Architecture: Offline Deterministic Compiler & Self-Hosted Rust Data Plane

Instead of wrapping raw REST endpoints in a thin runtime proxy, Arcverin separates design-time proof from production execution across a cryptographic trust boundary. Both Stage 3's offline fault simulator and Stage 5's native Rust data plane consume the same compiled capability contract and WorkerIr-eliminating drift between CI verification and live execution:



Architectural Plane	Core Binaries & Artifacts	Verified Security & Determinism Guarantee
Design & Compiler Plane (Stages 1-4)	arcverin, arcverin-studio, arcverin-qualify, WorkerIr (worker.ir.json), OpenAPI 1.2.0 Overlays, Arazzo 1.1	100% offline deterministic by default; fail-closed {value | unknown} slots reject TBD placeholders; zero-regression Overlay rollback; restrict-only Arcverin Policy Visual JDM (rejects JS functionNode).
Cryptographic Trust Boundary	arcverin-commissioning, host-locked .dwpack (X25519 + XChaCha20-Poly1305), Ed25519 DSSE Packs, SHA-256 RFC 3339 waivers	Dual-control Operator + Reviewer commissioning; DSSEv1 PAE Ed25519 integrity (arcverin pack verify exits code 3 on tamper); expiring waivers fail closed on any OpenAPI byte change.
Zero-Trust Data Plane (Stage 5)	arcverin-runner (MCP 2026-07-28), arcverin-runtime, arcverin-vault (Argon2id + XChaCha20-Poly1305) / HashiCorp Vault KV v2	Secrets inject strictly outside LLM context; external OPA PDP (--opa-url) operates restrict-only and fails closed on timeout/error; append-only SHA-256 hash-chained policy-journal.jsonl.
Enterprise Control Plane	PostgreSQL 15-18 RLS, OIDC SSO / SCIM 2.0, OpenAPI 3.2.1 Headless Control-Plane API & Stateless MCP (/api/v2/mcp)	FORCE ROW LEVEL SECURITY across 15 workspace-scoped tables with transaction-scoped set_config (SET LOCAL, PgBouncer safe); agents execute up to AR2 and are hard-blocked from self-approval.

2. Choosing Your MCP Mode Under 90% Prompt Caching: Curated Eager vs. Progressive 1-2-3

Prompt caching cuts repeated static input-token cost by up to 90%, changing the practical math of MCP tool loading. Compaction remains critical to avoid the >50-tool LLM selection accuracy cliff-but choosing the right arcverin-runner exposure mode avoids needless round-trip latency and prevents nested indirection ('hat on a hat') with native client-side Tool Search:

MCP Exposure Mode	Upfront Footprint & Latency	Best-Fit Architecture & Engineering Trade-Off
Raw 1:1 OpenAPI Dump (Baseline)	37,885-137,012 o200k_base tokens; 0 extra turns; degrades past ~50 tools	Anti-pattern for enterprise APIs: causes tool-selection accuracy drops and high Turn-1 cache-write + per-turn latency overhead even under 90% prompt caching.
Curated Eager (--mode eager) [Recommended Default]	223-502 tokens per goal tool; 0 extra discovery round-trips	Wins on both cached session cost and latency for bounded domains. Exposes clean goal schemas directly so frontier clients with native Tool Search (Claude Code, Codex, Copilot) index tools without nested meta-tool indirection.
Progressive 1-2-3 (--mode progressive)	~380-550 upfront tokens; +2 discovery turns (discover -> describe -> invoke)	Best when one agent must span massive multi-domain catalogs (dozens of curated tools or 150+ control-plane ops) inside custom or open-source agent harnesses lacking native client-side Tool Search.

- Honest ROI driver: Under 90% prompt caching, token bill savings alone shrink substantially compared to uncached rates. Production payback comes primarily from eliminating the >50-tool accuracy cliff, reducing security/engineering review hours, and preventing duplicate-billing or cross-tenant incidents.
- Payload-size discipline: Compaction trims tool-description schemas, not API response bodies; pair with bounded pagination caps (Pillar 4: Rate, Cost & Blast Radius) or two-lane Code Mode read filtering so bulky JSON responses do not flood context windows.

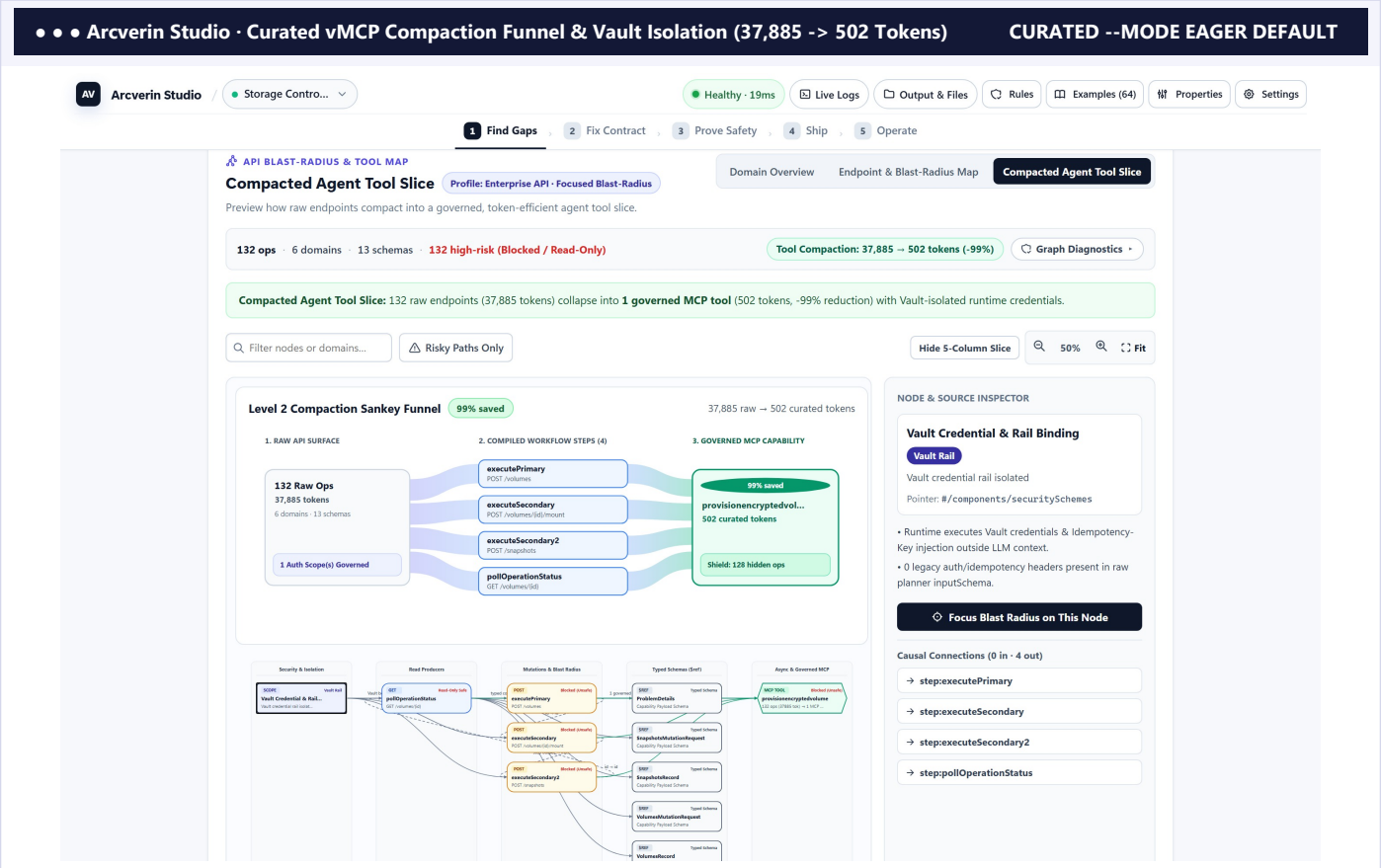


Figure 1: Curated vMCP collapses 132 raw Storage Control Plane operations into 1 outcome-focused capability (502 exact o200k_base tokens) with zero extra discovery round-trips and encrypted Vault binding.

3. Honest Production Boundaries & Distributed-Systems Reality

Credible architecture guides document where deterministic pre-runtime tooling ends and distributed backend engineering begins. Account for these five operational realities before deploying autonomous mutations:

Operational Reality	What Arcverin Proves	Required Production Engineering Practice
Map vs. Territory (Simulator vs. Live Service)	Offline qualification, Overlays & --target mock verify compiled contracts and client retry logic.	If a simulator reports a retry created one job, your own service/scheduler still proves how many jobs it actually created: pair mock runs with --target staging --read-only and customer-run --import receipts.
Read-Replica Lag & Multi-Provider Sagas	Proves single-client 504 pre-retry lookup invariants in deterministic simulation.	Ensure idempotency GET lookup paths read with strong read-after-write consistency (avoiding lagging replicas returning 404). Cross-provider Arazzo DAGs require explicit compensating steps rather than automatic 2PC.
Fail-Closed OPA Availability	arcverin-runner denies calls fail-closed on OPA timeout (--opa-timeout-ms, default 500 ms) or HTTP error.	Deploy OPA as a co-located sidecar pulling signed policy bundles rather than a remote single point of failure so transient network blips cannot halt mutating workflows.
CMK Signing & Platform Targets	Ed25519 DSSE packs with key-URI provenance metadata (cmk-key-provenance.json); native Windows x64 & Linux x86_64 glibc binaries.	Enterprise CMK resolves customer-managed Ed25519 seeds from your KMS/Vault secret pipeline rather than remote Azure Key Vault HSM signing RPC (matching AKV HSM's RSA/ECDSA curve set). macOS/ARM64 teams run Linux containers or CI runners.
Arazzo 1.1 & Schema Portability	Standard API/async steps export cleanly to Arazzo 1.1; x-arcverin-context enforces strict schema v1.1.0 (schemaVersion, accountableOwner, tenantBoundary, recovery).	Arcverin runtime guardrails travel in x-arcverin-dw-* extensions; third-party Arazzo runners execute standard steps while ignoring custom extensions unless enforced at your gateway.

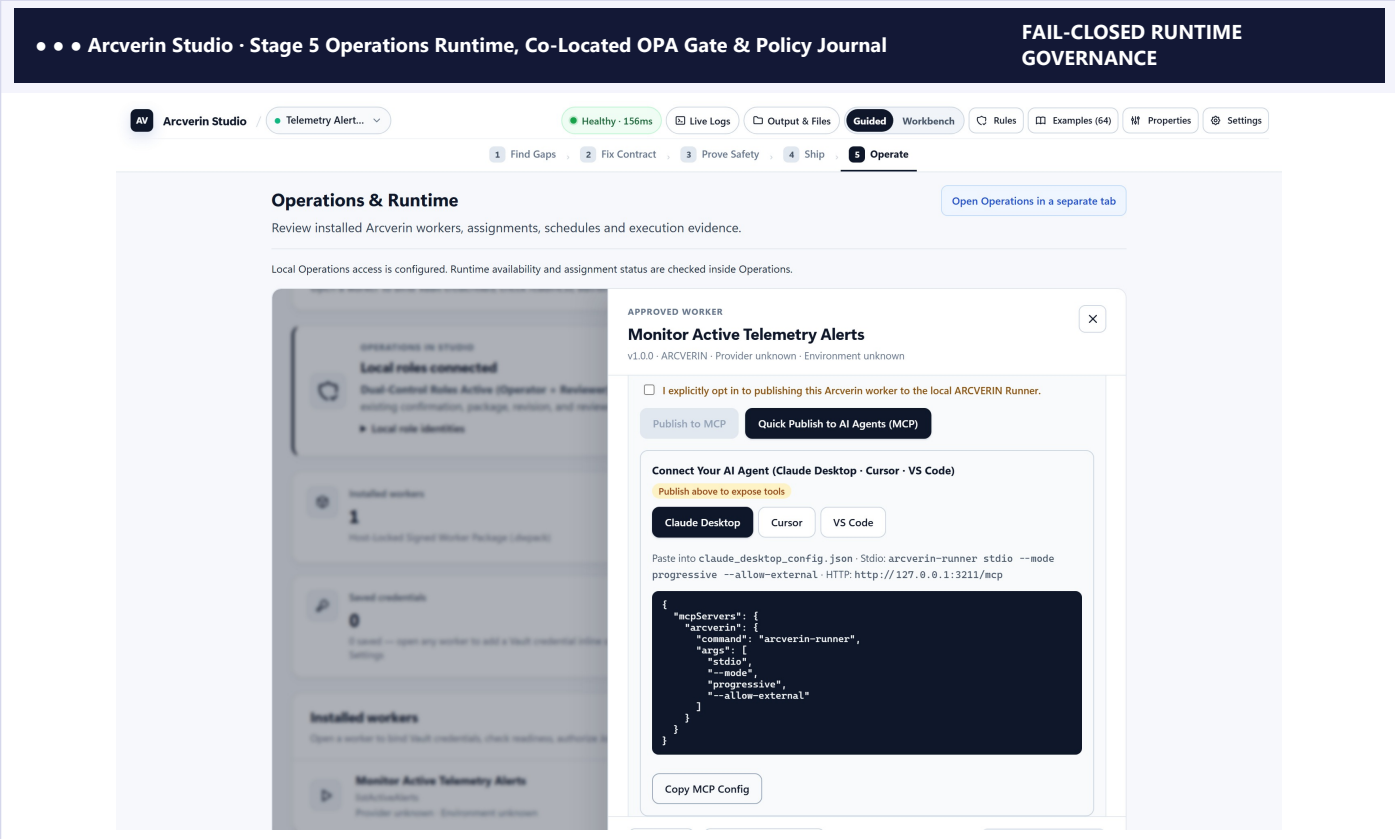


Figure 2: Stage 5 (5. Operate) Enterprise Operations console combining dual-control commissioning, Vault credential management, and fail-closed runtime execution governance.

4. 30-Minute Hands-On Buyer Verification Checklist (6 Offline Checks)

Do not take any vendor benchmark on faith. Start with a $99 one-time 1-Month Personal Pass (no auto-renewal) on an isolated workstation and execute these six reproducible checks against your own OpenAPI specification before unlocking a Team or Enterprise deployment:

- 1. Binary Provenance, Linkage & Catalog Parity: Verify SHA256SUMS.txt, measure binary size and Linux dynamic linkage (ldd ./arcverin-runtime), run 'arcverin rules list' to verify all 102 built-in rules (86 core + 11 authz/isolation/limits + 5 MCP token economy), and run 'arcverin validate --help' to inspect all 12 --fault profiles.
- 2. Offline Scale & OpenAPI 3.2 Benchmark: Disconnect outbound network access and run 'arcverin qualify scan <spec> --out ./out' plus 'arcverin mcp-economy <spec> --tokenizer o200k_base' on a 500+ operation internal spec (including OpenAPI 3.2 RFC 10008 QUERY operations if used) while recording peak RSS RAM and elapsed time.
- 3. Open-Standards Interoperability Drill: Compile a mutating capability and validate exported workflow.arazzo.json with external Arazzo validators, run 'opa test' against exported Rego + data files, and inspect --sarif output from 'arcverin gate' in GitHub Code Scanning.
- 4. Chaos Fault Proof vs. Static Linters: Run 'arcverin validate --capability <cap> --spec <spec> --target mock --fault commit-then-drop-response' to verify sideEffectCount=1 with pre-retry GET reconciliation vs. duplicate writes on naive retries, comparing findings against pure static OpenAPI linters.
- 5. Fail-Closed OPA & Journal Tamper Drill: Start 'arcverin-runner serve --mode eager --opa-url http://127.0.0.1:8181 --journal-path ./policy-journal.jsonl' against a local OPA sidecar; stop OPA mid-run to confirm fail-closed denial, then alter one byte in policy-journal.jsonl and confirm 'arcverin-runner verify-journal --journal-path ./policy-journal.jsonl' fails integrity verification.
- 6. Pre-Pilot Enterprise Architecture Review: Confirm PostgreSQL 15-18 FORCE ROW LEVEL SECURITY across 15 workspace tables + SET LOCAL PgBouncer compatibility, review KMS/Vault key-injection provenance, inspect exported gateway bundles (Azure APIM, Kong, Envoy, AWS Bedrock AgentCore), and review EULA Section 10 protections.

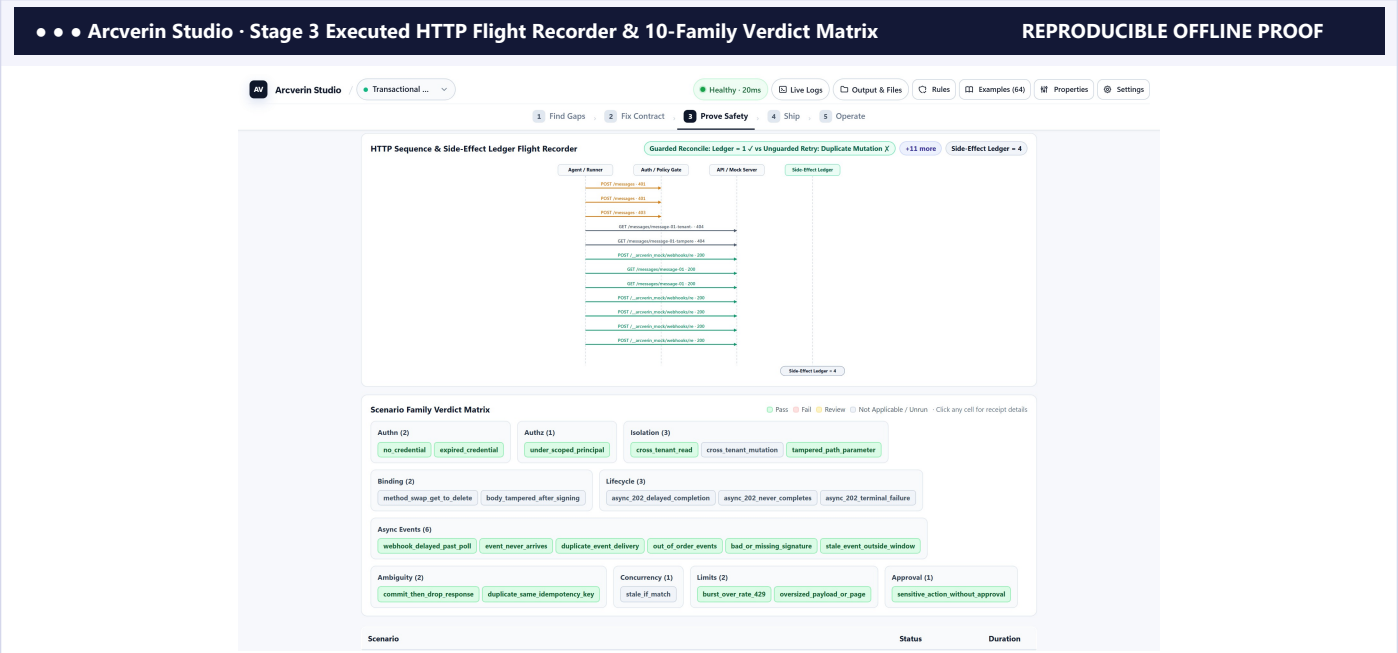


Figure 3: Stage 3 (3. Prove Safety) executing stateful mock validation across the 10-family Scenario Verdict Matrix with an exact HTTP Flight Recorder and side-effect ledger.

Start With a $99 30-Day Personal Pass & Download Synthetic Benchmark Specs at https://www.arcverin.com/ →

https://www.arcverin.com/

Engineering honesty note: Published token counts use downloadable vendor-neutral synthetic OpenAPI 3.1 fixtures (reference-benchmarks.zip) measuring MCP tool-description payloads, not guaranteed provider invoice cuts or third-party security certifications. Always verify behavior against your own staging backends (--target staging) before production release.