- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

  Formal Reviews & Dissents
  - Objection
  - Response
  - Follow-Up

ADR Clarification Record: ADR-OS-007

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

  Formal Reviews & Dissents
  - Objection
  - Response
  - Follow-Up

ADR Clarification Record: ADR-OS-008

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist
- Formal Reviews & Dissents

ADR Clarification Record: ADR-OS-009

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-010

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-011

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-012

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-013

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-014

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-015

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-016

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary

- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-017

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-018

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-019

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-020

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-021

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes

- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-028

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-029

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-030

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-031

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions
- Responses
- Formal Reviews & Dissents
- Additional Notes
- Traceability
- Distributed-Systems Protocol Compliance Checklist

ADR Clarification Record: ADR-OS-032

- Initial Clarification Draft (TBD)
- Assumptions & Constraints
- Dependencies & Risks
- Summary
- Clarification Questions

ADR Clarification Record: ADR-OS-001

Initial Clarification Draft (architect-1)

Assumptions & Constraints

- The OS persists `state.txt.ph` atomically between phases and retries failed writes.
- A healthy quorum of agent runners is available before any phase transition.
- All phase-transition pre-conditions are evaluated in a deterministic order to avoid race conditions.
- Distributed trace propagation (`trace_id`) is mandatory for every mutating action.

Dependencies & Risks

- Depends on ADR-OS-023 (idempotency keys) and ADR-OS-027 (vector clocks) for safe retries and event ordering.
- Requires Appendices A/B to remain the single source of truth for operational principles.
- Risk: If the distributed trace pipeline is unavailable, observability gaps may delay incident response.
- Risk: Mis-ordered vector-clock updates can deadlock the phase state machine.

Summary

The original ADR adopts a five-phase operational loop (ANALYZE → BLUEPRINT → CONSTRUCT → VALIDATE → IDLE) governed by `state.txt`. This clarification reiterates that the loop is **event-driven** and **idempotent**, and that every phase transition must emit an immutable audit-trail entry.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What is the fallback strategy if a phase transition times out due to a network partition?	architect-1	2025-06-28	OPEN	
2	Can a human override the automatic rollback triggered by validation failure?	architect-1	2025-06-28	OPEN	
3	How will the OS handle tasks that require jumping back to a previous phase (e.g., a validation failure requiring more construction)?	Hybrid_AI_OS	2025-06-27	OPEN	
4	What is the mechanism for a human to override a phase transition?	Hybrid_AI_OS	2025-06-27	OPEN	
5	How are partial or failed phase transitions detected and recovered, especially in distributed or partially available environments?	Hybrid_AI_OS	2025-06-27	OPEN	
6	What are the escalation and notification procedures if a phase transition is blocked or deadlocked?	Hybrid_AI_OS	2025-06-27	OPEN	
7	How does the OS ensure traceability and auditability of all phase transitions, including manual overrides?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Additional Notes

- Aligns with Theory of Constraints by exposing bottlenecks at each phase gate.
- Future iterations should define SLAs for phase-transition duration and recovery.

Traceability

- adr_source: ADR-OS-001
- trace_id: trace://auto-g69/resolve_placeholders
- vector_clock: vc://auto@69:0

Distributed-Systems Protocol Compliance Checklist

- [x] Idempotent updates supported
- [x] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

Formal Reviews & Dissents

<-- help test -->

ADR Clarification Record: ADR-OS-002

Initial Clarification Draft (architect-2)

Assumptions & Constraints

- The hierarchical planning store is implemented on a CP data store with single-writer semantics per artifact ID. *(Confidence: High; Self-Critique: write latency may increase under Raft leader change)*
- Each planning artifact embeds its **parent_id** to ensure bidirectional traceability. *(Confidence: Medium; Self-Critique: duplication risk if parent changes post-creation)*
- Supervisor Agent enforces **idempotent** creation of linkage edges to prevent duplicate child plans. *(Confidence: High)*
- Vector-clock field (`vc`) and `trace_id` are mandatory on every create/update mutation. *(Confidence: High; Self-Critique: tight coupling with tracing backend)*

Dependencies & Risks

- Depends on ADR-OS-024 (asynchronous hand-offs) to allow independent creation of Analysis & Initiative layers.
- Relies on ADR-OS-027 vector-clocks for causal linkage ordering; risk: clock mis-merge may break lineage. *Mitigation: Lamport fallback counter.*
- Hierarchical store downtime blocks new Execution Plans. *Mitigation: local cache + replay queue.*

Summary

ADR-OS-002 introduces a four-tier **Request**→**Analysis**→**Initiative**→**Execution** model. The clarification emphasises **immutable lineage**, idempotent link creation, and distributed consistency guarantees so that every Execution Plan can be traced back to its originating business Request.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	How are orphaned Execution Plans detected & repaired?	architect-2	2025-06-28	OPEN	
2	What SLA applies to linkage propagation across tiers?	architect-2	2025-06-28	OPEN	
3	Can an Initiative Plan spawn sibling Execution Plans in parallel without parent lock?	architect-2	2025-06-28	OPEN	
4	How will the system handle a `Request` that spawns multiple, independent `Initiative Plans`?	Hybrid_AI_OS	2025-06-27	OPEN	
5	What is the process for archiving or closing out a completed `Initiative Plan` and its children?	Hybrid_AI_OS	2025-06-27	OPEN	
6	How are planning artifact linkages (e.g., parent-child relationships) validated and repaired if broken?	Hybrid_AI_OS	2025-06-27	OPEN	
7	What is the recovery process if the planning artifact store becomes inconsistent or partially unavailable?	Hybrid_AI_OS	2025-06-27	OPEN	
8	How does the system ensure that context and traceability are preserved during asynchronous or concurrent plan creation?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Traceability

- adr_source: ADR-OS-002
- trace_id: trace://auto-g69/resolve_placeholders
- vector_clock: vc://auto@69:1

Distributed-Systems Protocol Compliance Checklist

- [x] Idempotent updates supported (linkage creation via idempotency key)
- [x] Message-driven integration points documented (Supervisor Agent events)
- [ ] Immutable audit-trail hooks attached (implementation in-progress, target g60)

Formal Reviews & Dissents

<-- help test -->

ADR Clarification Record: ADR-OS-003

Initial Clarification Draft (TBD)

Assumptions & Constraints

- The `EmbeddedAnnotationBlock` JSON structure will remain backward-compatible across future schema iterations.
- All text-editable file formats in the repository can safely embed comment-wrapped JSON without breaking primary runtime semantics.
- Agents executing CONSTRUCT and VALIDATE phases possess deterministic parsers that preserve exact annotation formatting (idempotent write-back).
- Parsing and serializing these annotation blocks adds negligible latency (<5 ms per artifact) to standard CI/CD workflows.
- The global registry (`os_root/global_registry_map.txt`) is considered the single authoritative index; any temporary divergence must be self-healing within one execution cycle.

Dependencies & Risks

- **Upstream ADRs:** Depends on ADR-OS-023 (Idempotency), ADR-OS-029 (Observability & Tracing), ADR-OS-032 (Canonical Models Registry).
- **Toolchain:** Relies on custom linters and CI hooks described in Appendix H to validate annotation integrity—failure or mis-configuration may allow malformed blocks to merge.
- **Distributed Coordination:** Concurrent agents updating the same artifact risk write conflicts; without optimistic-locking, last-writer-wins could corrupt JSON.
- **Schema Drift:** Rapid evolution of the annotation schema could strand legacy artifacts; mitigation includes version gating and automated migrations.
- **Human Error:** Manual edits to annotation blocks may introduce syntax errors that break parsing pipelines.

Summary

ADR-OS-003 mandates embedding a structured JSON metadata block—`EmbeddedAnnotationBlock`—at the top of every text-editable project artifact. This block captures purpose, authorship, dependencies, and traceability, enabling fully self-describing artifacts that support automated governance, distributed tracing, and architectural enforcement. The decision favors in-file annotations over external databases or sidecar files to maintain a single source of truth closely coupled with version control history.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What recovery process is defined if an `EmbeddedAnnotationBlock` becomes corrupted or partially deleted?	Hybrid_AI_OS	2025-06-27	OPEN	
2	How will concurrent annotation updates be orchestrated to avoid race conditions across distributed agents?	Hybrid_AI_OS	2025-06-27	OPEN	
3	What strategy will be employed for non-text artifacts (e.g., binaries) that cannot embed JSON blocks?	Hybrid_AI_OS	2025-06-27	OPEN	
4	How will schema version migrations be automated to guarantee backward compatibility?	Hybrid_AI_OS	2025-06-27	OPEN	
5	What audit trail fields (e.g., `trace_id`, `vector_clock`) are mandatory within each annotation update event?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Additional Notes

- Appendix A provides general assumption-surfacing guidelines applied here.
- Refer to Appendix H for CI/CD linter enforcement details ensuring presence and validity of annotation blocks. Failure cases discovered during internal DR drills should be back-ported to this clarification record.
- Future iterations should include a diagram (see ADR-OS-003 self-critique) illustrating annotation placement across representative file formats.

Traceability

- adr_source: ADR-OS-003
- trace_id: trace://auto-g69/resolve_placeholders
- vector_clock: vc://auto@69:2

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

Formal Reviews & Dissents

<-- help test -->

ADR Clarification Record: ADR-OS-004

Initial Clarification Draft (TBD)

Assumptions & Constraints

- The global event counter (g) is incremented atomically and never skipped—even under high concurrency or network partitions.
- Every write to a mutable OS Control File must perform a read-check-increment-write cycle on its version counter (v).
- Agents performing file mutations share a common optimistic-locking library that enforces v semantics uniformly across languages/runtimes.
- Vector clocks (ADR-OS-027) are available and correctly configured for workflows needing causal ordering beyond simple total ordering.
- The `state.txt` file housing g remains highly available (>99.9%) within the primary partition; minority partitions fall back to read-only mode.
- CI pipelines include linters that fail fast when they detect gaps or regressions in g or v semantics.

Dependencies & Risks

- **Upstream ADRs:** ADR-OS-027 (Logical Clocks), ADR-OS-028 (Partition Tolerance), ADR-OS-029 (Observability & Tracing).
- **Hot-Spot Contention:** Heavy writes to `state.txt` may throttle throughput. Mitigation: batching low-priority events or introducing sharded counters (future work).
- **Distributed Atomicity:** Achieving atomic increments for g in multi-node deployments requires consensus (e.g., Raft) or single-writer enforcement.
- **Stale Write Risk:** Any agent skipping the v check can overwrite newer data—guard rails include schema validators and pre-commit hooks.
- **Corruption Recovery:** If g/v values diverge or become corrupted, agents must enter `BLOCK_INPUT` state until a reconciliation procedure restores monotonicity.

Summary

ADR-OS-004 formalizes a dual-counter strategy for sequencing and safeguarding OS operations: a monotonic Global Event Counter (g) provides total ordering and unique IDs for significant events, while a per-file Version Counter (v) enforces optimistic locking to prevent stale writes. Together they create a lightweight yet powerful audit trail and concurrency-control mechanism suitable for distributed agent collaboration.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What consensus or locking mechanism ensures atomic g increments across multiple OS instances?	Hybrid_AI_OS	2025-06-27	OPEN	
2	What automated rollback or repair process exists if a gap (skipped g) or duplicate is detected in production?	Hybrid_AI_OS	2025-06-27	OPEN	
3	How will high-frequency event generation avoid write contention on `state.txt` without sacrificing strict ordering?	Hybrid_AI_OS	2025-06-27	OPEN	
4	Under what conditions can minority partitions safely continue read-write operations, or must they remain read-only until reconciliation?	Hybrid_AI_OS	2025-06-27	OPEN	
5	Which log/trace fields are mandatory to correlate g, v, and `trace_id` for complete observability?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Additional Notes

- Appendix H outlines CI/CD enforcement hooks that verify monotonicity of g and correctness of v increments.
- Appendix B details error-handling roles; any detected counter anomaly must trigger the **Partition Reconciliation** playbook.
- A future ADR may introduce sharded or hierarchical counters to alleviate the single-hot-spot nature of g.

Traceability

- adr_source: ADR-OS-004
- trace_id: trace://auto-g69/resolve_placeholders
- vector_clock: vc://auto@69:3

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

Formal Reviews & Dissents

<-- help test -->

ADR Clarification Record: ADR-OS-005

Initial Clarification Draft (TBD)

Assumptions & Constraints

- A valid `haios.config.json` file is guaranteed at repo root and loaded before any OS action.
- All paths declared in the config remain within repository boundaries to preserve portability.
- The filesystem supports nesting and naming conventions defined (case-sensitive where required).
- Config schema evolution is strictly backward-compatible or accompanied by an auto-migration script.
- The OS operates with least-privilege FS permissions; missing write access to configured paths is treated as fatal.

Dependencies & Risks

- **Upstream ADRs:** ADR-OS-032 (Canonical Models Registry) governs naming conventions; ADR-OS-029 (Observability) ensures config load is traced.
- **Single Point of Failure:** Corrupted or missing `haios.config.json` prevents startup; mitigated via schema validation and fallback wizard.
- **User Misconfiguration:** Incorrect path mappings could leak OS control files into versioned app code—CI linter halts merge.
- **Cross-Platform Variance:** Path separators and case sensitivity differ between OSes; mitigated via runtime normalization utilities.
- **Config Drift:** Manual edits may diverge from actual directory layout; scheduled VALIDATE jobs reconcile and raise issues.

Summary

ADR-OS-005 adopts a configuration-driven directory structure, centralizing all operational paths in `haios.config.json`. This makes Hybrid_AI_OS portable, predictable, and easy to integrate into diverse project layouts while retaining clear separation between OS internals and project artifacts.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What versioning strategy is defined for `haios.config.json` to support breaking schema changes?	Hybrid_AI_OS	2025-06-27	ANSWERED	Embed a top-level `config_schema_version` (semver) field; CI blocks if version bump lacks migration script.
2	How should the OS react to extra, unrecognized keys in the config file—ignore or fail fast?	Hybrid_AI_OS	2025-06-27	ANSWERED	Linter issues WARN for unknown keys in minor versions; MAJOR versions may elevate to ERROR per strict mode.
3	Is there a CLI bootstrap command to regenerate a default directory scaffold if the config is missing?	Hybrid_AI_OS	2025-06-27	ANSWERED	`haios init --scaffold` generates default paths idempotently and writes validated config file.
4	What guardrails prevent user-supplied paths from escaping the repository root (path traversal)?	Hybrid_AI_OS	2025-06-27	ANSWERED	Loader canonicalizes paths and verifies they reside under repo root using `Path.resolve().is_relative_to(root)`; CI blocks traversal attempts.
5	How will multi-repository or mono-repo setups override or extend the single config paradigm?	Hybrid_AI_OS	2025-06-27	ANSWERED	Support layered override files (`haios.config.local.json`, workspace-level config) merged via deep-merge with precedence; union validated against master schema.

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	architect-1	2025-06-27	1		Use semver field plus migration script gating.
2	architect-1	2025-06-27	2		Unknown keys trigger warnings unless strict mode; future-proofing configs.
3	architect-1	2025-06-27	3		Introduced `haios init --scaffold`; safe rerun.
4	architect-1	2025-06-27	4		Path canonicalization prevents escapes; linter guards.
5	architect-1	2025-06-27	5		Layered configs enable multi-repo; deep-merge with precedence.

Additional Notes

- Appendix C (Scaffold Definition Template) outlines default directory layout used by the bootstrap command.
- Appendix A's assumption-surfacing checklist guided the expanded assumption block above.
- Future work: provide a visual directory tree diagram in docs/source to complement textual spec.

Traceability

- adr_source: ADR-OS-005
- trace_id: trace://auto-g69/resolve_placeholders
- vector_clock: vc://auto@69:4

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

Formal Reviews & Dissents

<-- help test -->

Author: architect-2 **Date:** 2025-06-27

Objection

| Concern # | Related Q# | Summary of Objection | Suggested Follow-up | |----------|----------|----------|----------| | 1 | 1 | Sole reliance on a semver field is fragile; without explicit JSON Schema `$id` and `$schema` references, tooling may mis-interpret breaking changes. | Augment config with `$schema` URL pointing at tagged schema version; update CI to fetch and validate against that exact revision. | | 2 | 2 | Treating unknown keys as warnings allows silent drift; strict mode should be default to prevent typo-driven misconfig. | Flip default to strict validation; allow opt-in `allowUnknownKeys` in config for extension points, guarded by feature flag. | | 3 | 3 | `haios init --scaffold` may overwrite existing customized paths, risking data loss. | Add `--dry-run` and interactive diff output; require explicit `--force` to modify existing config. | | 4 | 4 | Path canonicalization misses symlink traversal edge-cases; malicious symlinks could bypass root check. | Resolve symlinks and verify inode ancestry; optionally block symlink usage in control-file directories. | | 5 | 5 | Layered override approach may yield nondeterministic results in mono-repo with nested workspaces; precedence rules unclear. | Document deterministic precedence order (workspace > repo > default) and add linter to detect override cycles. |

Author: architect-1 **Date:** 2025-06-27

Response

| Concern # | Disposition | Mitigation / Next Action | |----------|----------|----------| | 1 | ACCEPT | Embed `$schema` URL referencing semver-tagged schema revision; CI validator locked to that URL. Update config generator accordingly. | | 2 | ACCEPT PARTIAL | Default to strict validation; introduce optional `allowUnknownKeys` flag behind feature gate for extension points. Add drift-detection unit tests. | | 3 | ACCEPT | Enhance `haios init --scaffold` with `--dry-run`, interactive diff, and `--force` flags to safeguard customized paths. | | 4 | ACCEPT | Resolver will follow symlinks and assert final inode under repo root; block symlinks in OS control directories. Add security test harness. | | 5 | ACCEPT | Document precedence order (workspace > repo > default) and implement linter rule to detect override cycles. |

ADR Clarification Record: ADR-OS-006

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Every `Scaffold Definition` JSON validates against the schema in docs/Schema/scaffold_definition_schema.md.
- All boilerplate assets referenced exist within `project_templates/` at scaffold time.
- The OS has adequate write permissions for all target paths in `project_workspace/`.
- Scaffold execution is idempotent; running the same definition twice produces no unintended duplicates.
- Each scaffolded artifact automatically receives a complete `EmbeddedAnnotationBlock` based on template metadata.
- Template placeholder syntax remains stable to avoid breaking older Scaffold Definitions.

Dependencies & Risks

- **Upstream ADRs:** ADR-OS-003 (Annotations) ensures metadata embedding; ADR-OS-032 (Canonical Models Registry) defines template compliance.
- **Template Drift:** Updates to shared templates can unintentionally alter existing code bases—CI diff checks required.
- **Partial Failures:** Mid-process interruption may leave inconsistent state; mitigation via transaction-like rollback or resume logic.
- **Complex Placeholder Logic:** Advanced replacement rules increase maintenance overhead and risk of mis-rendering files.
- **Versioning:** Lack of scaffold version pinning could break reproducibility; recommend semantic versioning of Scaffold Definitions.

Summary

ADR-OS-006 introduces a schema-based scaffolding system driven by `Scaffold Definition` JSON files. It orchestrates the automated creation of new components using assets from `project_templates/`, injecting rich annotations so artifacts are self-describing from inception. This accelerates development, enforces best practices, and guarantees consistent structure across the project.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	How are Scaffold Definitions versioned and migrated without breaking existing components?	Hybrid_AI_OS	2025-06-27	ANSWERED	Each definition carries `scaffold_def_version` (semver) + `template_version`. CI requires migration script or backward-compat flag when major bump detected.
2	What rollback mechanism exists if scaffolding fails midway through artifact generation?	Hybrid_AI_OS	2025-06-27	ANSWERED	Scaffold executor writes to temp staging dir; commits atomically on success else auto-rolls back and logs failure event with trace_id.
3	How will placeholders and conditional logic within templates be validated to prevent runtime syntax errors?	Hybrid_AI_OS	2025-06-27	ANSWERED	Pre-execution static analysis & unit compile of rendered templates; CI breaks if placeholder unresolved or syntax lint fails.
4	Can end-users override default templates while still passing CI scaffold integrity checks?	Hybrid_AI_OS	2025-06-27	ANSWERED	Users may reference custom templates via `template_registry_path`; linter validates schema conformance and required annotation fields.
5	What metrics will be collected to monitor scaffold usage, success rate, and drift over time?	Hybrid_AI_OS	2025-06-27	ANSWERED	Telemetry captures `scaffold_id`, duration, success/fail, drift hash, agent_id; aggregated in metrics dashboard with weekly report.

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	architect-1	2025-06-27	1		Semver + migration gating ensures safe evolution; minor/patch auto-compatible.
2	architect-1	2025-06-27	2		Temp staging directory enables atomic commit/rollback.
3	architect-1	2025-06-27	3		Static analysis + compile test catches placeholder issues pre-merge.
4	architect-1	2025-06-27	4		Custom templates allowed but must pass same schema and annotation checks.
5	architect-1	2025-06-27	5		Telemetry dashboard provides visibility on scaffold health & drift.

Additional Notes

- Appendix C details Scaffold Definition Template guidelines referenced here.
- Appendix F outlines testing guidelines; scaffold-generated tests should conform automatically.
- Future diagram needed to illustrate scaffold workflow and rollback paths.

Traceability

- adr_source: ADR-OS-006
- trace_id: trace://auto-g69/resolve_placeholders
- vector_clock: vc://auto@69:5

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented

- [ ] Immutable audit-trail hooks attached

Formal Reviews & Dissents

<-- help test -->

Author: architect-2 **Date:** 2025-06-27

Objection

| Concern # | Related Q# | Summary of Objection | Suggested Follow-up | |----------|----------|----------|----------| | 1 | 1 | Relying on separate `scaffold_def_version` and `template_version` fields could diverge; unclear which drives CI gating. | Adopt single authoritative `schema_version` embedded in both definition and template header; CI cross-check for match. | | 2 | 2 | Temp staging dir rollback may leave residual orphan files if process killed mid-rename. | Use transactional FS library with journal + checksum verification; add cleanup sentinel job. | | 3 | 3 | Static analysis compile test assumes language-specific compilers exist for all templates; slows pipeline. | Provide language-agnostic placeholder linter first, compile tests optional per template tag. | | 4 | 4 | Allowing custom templates risks fragmentation and quality drift. | Require central registry approval + semantic diff review before custom template allowed in CI. | | 5 | 5 | Telemetry fields omit `template_version` and do not track failed placeholder resolutions count. | Extend metrics to include `template_version`, `placeholder_error_count`, and `rollback_flag`. |

Author: architect-1 **Date:** 2025-06-27

Response

| Concern # | Disposition | Mitigation / Next Action | |----------|----------|----------| | 1 | ACCEPT | Replace dual fields with single `schema_version` embedded in both Scaffold Definition and template header; CI diff check enforces equality. Provide migration script to auto-converge legacy records. | | 2 | ACCEPT | Integrate transactional FS lib with journaling & checksum; add watchdog cleanup job to purge residual staging artefacts on startup. | | 3 | ACCEPT PARTIAL | Placeholder linter mandatory for all templates; compile tests gated by optional `requiresCompileTest` flag in template metadata to keep pipeline lean. Cache compiled artifacts to reduce overhead. | | 4 | ACCEPT PARTIAL | Custom templates allowed only after central registry review & semantic diff approval; interim experimentation permitted behind `experimentalTemplate` flag not deployable to production. | | 5 | ACCEPT | Extend telemetry to capture `template_version`, `placeholder_error_count`, `rollback_flag`, and emit to metrics pipeline. |

Author: architect-2 **Date:** 2025-06-27

Follow-Up

| Concern # | Status After Mitigation | Additional Commentary | |----------|----------|----------| | 1 | SATISFIED | Confirm migration script includes dry-run diff mode and adds `schema_version` to template header generator. | | 2 | PENDING IMPLEMENTATION | Await integration of transactional FS library; please deliver design doc outlining journal format and cleanup sentinel logic. | | 3 | SATISFIED | Placeholder linter fallback acceptable; request compile-test coverage metric to monitor opt-in adoption. | | 4 | PARTIAL ACCEPTANCE | Experimental flag strategy valid; require sunset date review every 2 sprints to prevent perpetual experimental status. | | 5 | SATISFIED | Telemetry additions suffice; ensure dashboard includes trend line on placeholder errors over last 30 days. |

ADR Clarification Record: ADR-OS-007

Initial Clarification Draft (TBD)

Assumptions & Constraints

- A hardened, sandboxed runtime (e.g., Docker-in-Docker) is provisioned for every Testing Agent execution.
- Test scripts and results conform to the schema in docs/Schema/test_artifact_schema.md and are immutable once published.
- Validation Agent has read-only access to result artifacts and cannot alter them.
- Flaky test detection logic (three consecutive passes required) is enforced before a result is considered trustworthy.
- All test executions are tagged with `trace_id` and correlated g value for observability.

Dependencies & Risks

- **Infrastructure:** Requires reliable container orchestration or VM provisioning to isolate tests.
- **Artifact Bloat:** Storing signed test results for every run can increase repository size; mitigation via artifact storage offloading with hash pointers.
- **Flaky Tests:** Repeated failures/pass cycles can delay pipeline; statistical analysis jobs necessary.
- **Security:** Compromised Testing Agent could falsify results; require attestation and cryptographic signing.
- **Time Budget:** Comprehensive test suites may slow down CI; need parallel execution strategy.

Summary

ADR-OS-007 establishes an evidence-based testing lifecycle with segregation of duties: Coding Agent writes tests, Testing Agent executes in a trusted sandbox, and Validation Agent audits signed results. This ensures objective quality signals and guards against self-reporting or hallucinated success.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | Which cryptographic mechanism (e.g., Sigstore, GPG) will sign Test Results Artifacts? | Hybrid_AI_OS | 2025-06-27 | ANSWERED | Adopt Sigstore/cosign with OpenID Connect workload identity; fall back to project GPG keypair if cosign unavailable. Keys stored in Vault and rotated quarterly. | | 2 | What retention policy governs historical test result artifacts to manage storage footprint? | Hybrid_AI_OS | 2025-06-27 | ANSWERED | Retain signed results for last 200 successful pipelines and all failures for 1 year; older successes are GC'd after S3 archival snapshot with hash pointer kept. | | 3 | How are environment-specific variables (DB credentials, secrets) injected securely into test sandboxes? | Hybrid_AI_OS | 2025-06-27 | ANSWERED | Inject via ephemeral Vault tokens scoped to test duration using Kubernetes Secrets or Docker env masking; tokens auto-revoked post-run. No secrets persisted in result artifacts. | | 4 | What heuristics define and quarantine flaky tests for further investigation? | Hybrid_AI_OS | 2025-06-27 | ANSWERED | Flag test as flaky if <70% pass rate over last 10 runs or if fails/passes in 3 alternating runs; quarantined to separate job and marked `flaky=true` until stability proven. | | 5 | Can the Testing Agent execute destructive integration tests that mutate shared resources, and how is isolation ensured? | Hybrid_AI_OS | 2025-06-27 | ANSWERED | Destructive tests allowed only in ephemeral namespace or disposable infrastructure (Docker network, K8s namespace, test DB snapshot). Agent must declare `destructive=true`; sandbox wiped post-run. | | 6 | How is the "trustworthiness" of the `Testing Agent`'s execution environment technically enforced? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 7 | What is the exact schema for the `Test Results Artifact`? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | architect-1 | 2025-06-27 | 1 | | Sigstore cosign signatures with OIDC, GPG fallback, quarterly key rotation. | | 2 | architect-1 | 2025-06-27 | 2 | | Keep 200 latest successes + all failures 1yr; older archived to S3. | | 3 | architect-1 | 2025-06-27 | 3 | | Ephemeral Vault tokens injected via secrets engine; auto-revoked. | | 4 | architect-1 | 2025-06-27 | 4 | | Flaky if <70% pass/10 runs or 3 alternating; quarantine job. | | 5 | architect-1 | 2025-06-27 | 5 | | Destructive tests isolated in disposable namespace; flagged `destructive=true`. |

Additional Notes

- Appendix F (Testing Guidelines) elaborates on AAA structure and artifact schemas.
- Appendix H CI policy requires signed test results before merge.
- Potential future enhancement: results notarization via blockchain for tamper-evidence.

Traceability

- adr_source: ADR-OS-007
- trace_id: trace://auto-g69/resolve_placeholders
- vector_clock: vc://auto@69:6

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported

- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

Formal Reviews & Dissents

<-- help test -->

Author: architect-2 **Date:** 2025-06-27

Objection

| Concern # | Related Q# | Summary of Objection | Suggested Follow-up | |----------|----------|----------|----------| | 1 | 1 | Reliance on internet-based OIDC for Sigstore (`cosign`) breaks in air-gapped or high-security environments; fallback to project GPG keys may lack hardware-backed CA trust. | Provide offline signing workflow using YubiHSM or PKCS#11-compatible HSM; document procedure for air-gapped pipelines and ensure signature format parity with cosign. | | 2 | 2 | Retaining 200 successful pipelines can still reach >10 GB in large test suites; unclear GC strategy for failure artifacts beyond 1 year. | Introduce size-based retention cap (e.g., 50 GB) with LRU policy; compress artifacts (zstd) and move failures to cold-storage tier after 90 days, delete after SLA. | | 3 | 3 | Vault token injection may leak secrets via env variables visible in process lists or crash dumps. | Use file-based secret mounts with `tmpfs` + `chmod 0400`, and encrypt crash dumps; add agent hook to wipe memory after run. | | 4 | 4 | Fixed flaky heuristic (<70% pass) may misclassify sporadic integration flakes; could generate false positives & CI noise. | Implement adaptive heuristic using EWMA of failure rate; require sustained low confidence before quarantine; escalate to Test Stability Council review. | | 5 | 5 | Destructive tests in disposable namespaces still risk shared infra quota exhaustion and noisy neighbors. | Schedule destructive tests in isolated runner pool with resource quotas & budget guardrails; add pre-flight capacity check and auto-throttle queue. |

Author: architect-1 **Date:** 2025-06-27

Response

| Concern # | Disposition | Mitigation / Next Action | |----------|----------|----------| | 1 | ACCEPT | Provide offline signing pathway via PKCS#11-compatible HSM (e.g., YubiHSM). Author script `haios-sign-offline` that mirrors cosign payload format. Update CI templates for air-gapped mode and add parity verification test. | | 2 | ACCEPT | Enforce dual retention caps: 50 GB max size and 200 pipelines, whichever first. Compress artifacts with zstd; move failure artifacts >90 days to cold storage, purge after SLA. Implement GC job. | | 3 | ACCEPT | Switch to `tmpfs` file-mount secrets with 0400 perms; mask env variables; patch agent to scrub memory on exit and encrypt crash dumps via AES-GCM with ephemeral key. | | 4 | ACCEPT PARTIAL | Replace static threshold with EWMA-based flake score plus confidence band. Quarantine only when score remains <0.7 for 3 consecutive windows; route to Stability Council. Implement metric and dashboard. | | 5 | ACCEPT | Create dedicated destructive-test runner pool with strict CPU/memory quotas and budget guardrails; add pre-flight capacity check and auto-throttle queue. |

Author: architect-2 **Date:** 2025-06-27

Follow-Up

| Concern # | Status After Mitigation | Additional Commentary | |----------|----------|----------| | 1 | SATISFIED | Please run interoperability test ensuring `haios-sign-offline` signatures verify with standard cosign tooling; deliver report in next sprint review. | | 2 | PENDING IMPLEMENTATION | GC job design looks good; request dry-run simulation to validate retention math and cold-storage transfer latency. | | 3 | SATISFIED | Ensure secret-scrub test harness includes deliberate crash to inspect memory dump encryption path. | | 4 | PARTIAL ACCEPTANCE | EWMA approach acceptable; re-evaluate threshold configs after 30-day trial to calibrate false negatives. | | 5 | SATISFIED | Provide Grafana dashboard for destructive-runner pool resource consumption and auto-throttle events. |

ADR Clarification Record: ADR-OS-008

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Human supervisors value narrative Markdown reports and will review them at least once per sprint.
- Report generation time (<30s) does not impede CI/CD throughput.
- All reports are stored under `docs/reports/` and indexed in `global_registry_map.txt`.
- Templates for Analysis, Validation, and Progress Review reports are version-controlled and backward-compatible.
- Each report embeds `trace_id`, g snapshot, and `vector_clock` for linkage to underlying events.

Dependencies & Risks

- **Template Drift:** Divergent report outlines across versions can confuse readers; mitigation via semantic version tags and changelogs.
- **Report Fatigue:** Excessive frequency leads to neglect; default cadence configurable via `haios.config.json`.
- **Synthesis Quality:** Poor NLP summarization may misrepresent data; manual spot checks or LLM ensemble evaluation recommended.
- **Distributed Generation:** Concurrent agents might generate conflicting reports; v optimistic locking on report files enforced.
- **Storage Bloat:** Historical reports accumulate; archiving policy moves older reports to cold storage.

Summary

ADR-OS-008 mandates OS-generated, human-readable Markdown reports (Analysis, Validation, Progress Review) that narratively synthesize system state, reasoning, and evidence. Reports are fully traceable via embedded identifiers and follow standardized outlines to ensure clarity and comparability over time.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|---|---|---|---|---| | 1 | What is the expected default cadence for Progress Review reports to balance insight vs. fatigue? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How will report templates be migrated when outline changes introduce or remove sections? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Can supervisors annotate reports with inline feedback that feeds back into AI learning loops? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | What metrics will track report consumption (e.g., time-to-read, section scroll depth) to inform improvements? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How are cross-report inconsistencies detected and resolved when multiple agents produce overlapping narratives? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 6 | What are the specific schemas/outlines for each of the three report types, and how are they versioned and evolved? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 7 | How can a human or agent trigger an on-demand `Progress Review`, and what is the audit trail for such triggers? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 8 | What mechanisms will be in place to detect and mitigate low-quality, formulaic, or "gamed" self-assessments in reports? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 9 | How does the system handle report generation, consistency, and traceability in distributed or partitioned environments? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 10 | What is the process for updating, archiving, or deprecating report templates as project requirements evolve? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|---|---|---|---|---| | 1 | *placeholder* | | | | |

Additional Notes

- Appendix E (Reporting & Reviews) contains draft outlines; to be promoted to formal schema docs.
- README section on observability explains linking reports to traces.
- Future enhancement: interactive dashboards summarizing reports across time windows.

Traceability

- adr_source: ADR-OS-008
- trace_id: trace://auto-g69/resolve_placeholders
- vector_clock: vc://auto@69:7

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

Formal Reviews & Dissents

<-- help test -->

ADR Clarification Record: ADR-OS-009

Initial Clarification Draft (TBD)

Assumptions & Constraints

- File-system latency remains acceptable (<20 ms) for reading/writing individual issue files even at 10k+ issues.
- `issue_<g>.txt` adheres to docs/Schema/issue_schema.md, ensuring forward compatibility via `schema_version` field.
- Summary generation is event-driven; a hook updates initiative and global summaries atomically after each issue mutation.
- OS agents hold exclusive write locks on issue files during edits to avoid race conditions.
- Global summary size is capped; archived issues roll over into snapshot files.

Dependencies & Risks

- **Consistency Complexity:** Tri-level synchronization may introduce edge-case inconsistencies; mitigated with transactional updates and retry logic.
- **Scale Limits:** Large enterprise projects may outgrow file-based approach; long-term roadmap includes optional DB-backed store.
- **Merge Conflicts:** Concurrent branch edits to summary files could cause git conflicts; CI linter auto-resolves via deterministic ordering.
- **Schema Drift:** Evolution of issue schema requires migration tooling to patch historical files.
- **Visibility Gaps:** If summaries fail to update, cockpit dashboards show stale data; health checks monitor last-updated timestamps.

Summary

ADR-OS-009 formalizes a hierarchical, file-based issue tracking system comprising individual `issue_<g>.txt` records, initiative-level summaries, and a root-level global summary. This balances granular traceability with performant oversight for supervisors and agents.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | How will cross-initiative issue dependencies be represented—link arrays or separate relationship files? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | What specific locking or optimistic versioning strategy prevents concurrent edits to the same summary file? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Is there a garbage-collection or archiving policy for resolved issues beyond a certain age or count? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | Can issues be auto-synchronized with external trackers (GitHub issues) while retaining file-based SSOT? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | What dashboards or CLI commands will visualize issue metrics (age, severity, area) using summary data? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix D schema directory lists issue and summary schemas.
- Appendix B operational roles define ownership for triaging and updating issues.
- Future enhancement: automatic dependency graph generation for complex issue webs.

Traceability

- adr_source: ADR-OS-009
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-010

Initial Clarification Draft (TBD)

Assumptions & Constraints

- All `_locked*` boolean fields are validated by schema and default to `false`.
- Agents must check lock status before any mutating write; violation triggers automatic BLOCKER issue.
- Override requests are executed only via `REQUEST` artifacts signed by supervisor key.
- Distributed writes to lock fields are serialized through optimistic `v` counters to prevent split-brain.
- Lock metadata changes (<1 KB) keep state.txt size within performance budget.

Dependencies & Risks

- **Rigidity Risk:** Over-locking freezes evolution; governance board reviews lock additions.
- **Override Latency:** Human approval may slow urgent fixes; emergency bypass protocol TBD.
- **Partial Update:** Crash mid-write could leave stale lock state; atomic write ensured via temp-file swap.
- **Schema Sprawl:** Proliferation of lock types complicates agent logic; central enum registry proposed.
- **Monitoring Gaps:** Lack of dashboard for active locks may cause blind spots.

Summary

ADR-OS-010 introduces granular `_locked*` flags in schemas and annotations to safeguard immutable decisions. Agents encountering a `true` lock must halt, raise a BLOCKER issue, and await explicit supervisor-approved override, ensuring architectural integrity.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What signing mechanism authenticates override `REQUEST` artifacts?	Hybrid_AI_OS	2025-06-27	OPEN	
2	Will there be a namespace/schema for different lock types to avoid ambiguity?	Hybrid_AI_OS	2025-06-27	OPEN	
3	How can read-only minority partitions continue non-mutating work without lock conflicts?	Hybrid_AI_OS	2025-06-27	OPEN	
4	What telemetry will surface counts of active locks and recent override events?	Hybrid_AI_OS	2025-06-27	OPEN	
5	Are composite locks (affecting multiple fields atomically) supported, and how represented?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix B outlines error-handling escalation paths referenced here.
- Schema directory will host Locking Fields spec v1.0.
- Future enhancement: UI cockpit panel for lock status and override workflow.

Traceability

- adr_source: ADR-OS-010
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-011

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Retry policy is global: 3 attempts with exponential backoff starting at 5s.
- `FAILED` tasks automatically create `issue_<g>` and push an entry to `human_attention_queue.txt`.
- Remediation plans are labeled `remediation_<g>` and inherit context via `linked_issue_ids`.
- Validation Agent verifies remediation success before closing original issue.
- Failure logging includes stack trace, parameters, and environment snapshot for reproducibility.

Dependencies & Risks

- **Queue Overload:** Excessive failures could flood human_attention_queue; dashboard prioritization required.
- **Issue Proliferation:** Many small remediation plans may clutter history; quarterly cleanup policy recommended.
- **Misclassification:** Network partitions may mark tasks FAILED instead of BLOCKED; heuristic checks needed.
- **Retry Storms:** Misconfigured backoff could saturate resources; circuit breaker guards in executor.
- **Human Bottleneck:** Limited supervisor availability slows remediation; consider rota schedule.

Summary

ADR-OS-011 defines a "Log, Isolate, Remediate" strategy: after exceeded retries, tasks mark as FAILED, open an Issue, escalate to human queue, and spawn a dedicated Remediation execution plan to restore progress—all actions traced end-to-end.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What default exponential backoff formula is used (factor, jitter)?	Hybrid_AI_OS	2025-06-27	OPEN	
2	Can supervisors override retry thresholds per task type in config?	Hybrid_AI_OS	2025-06-27	OPEN	
3	How is partial success (some checklist items pass) represented before marking FAILED?	Hybrid_AI_OS	2025-06-27	OPEN	
4	What SLA exists for responding to human_attention_queue entries?	Hybrid_AI_OS	2025-06-27	OPEN	
5	Are automated diagnostics (logs, core dumps) attached to the Issue artifact?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix B operational principles outline escalation workflow.
- Appendix F testing guidelines cover flaky test detection which feeds into failure classification.
- Future enhancement: ML-driven triage suggestions for remediation tasks.

Traceability

- adr_source: ADR-OS-011
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-012

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Agent registry index (`agent_registry.txt`) is loaded into memory cache at startup with TTL 60s.
- New agent cards must pass schema validation (`agent_card_schema.md`) before being referenced in registry.
- Only Supervisor agent can mutate registry; all changes signed and include g event ID.
- Registry modifications are atomic: write temp file then move.
- Max agents supported in CP mode: 500; beyond triggers sharded registries roadmap.

Dependencies & Risks

- **Single Index Hotspot:** Frequent reads may cause contention; caching mitigates but eventual consistency lag risk.
- **Corruption:** Partial write could orphan card; recovery script scans and repairs inconsistencies.
- **Privilege Escalation:** Compromised Supervisor could alter agents; multifactor approval or commit-signing required.
- **Card Bloat:** Overly verbose agent history sections may slow parsing; consider log rotation field.
- **Partition Scenario:** Minority partitions operate with stale registry; tasks requiring missing persona are BLOCKED.

Summary

ADR-OS-012 designs a dynamic "Index + Individual File" agent management system: a central registry lists all persona IDs and paths, while detailed Agent Cards capture configuration, capabilities, and status. This allows runtime addition, update, or retirement of specialized agents without OS restart.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|---------|---------|-----|-------|-----------------| | 1 | What hashing or checksum verifies Agent Card integrity on load? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are long-lived agent state (e.g., fine-tuned weights) referenced or stored relative to card? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Can the registry support soft-deleting agents for historical audit while preventing assignment? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | What is the policy for rolling back a faulty agent card deployment? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How will sharding strategy evolve when agent count exceeds single index capacity? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|------------|------|-----------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix G Frameworks Registry ties into persona capability tags.
- Cockpit UI roadmap includes live agent health dashboard fed by registry events.
- Future enhancement: event-stream subscription model replacing polling for registry updates.

Traceability

- adr_source: ADR-OS-012
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-013

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Readiness checks run within 2s wall-clock to minimize plan latency.
- Checklist includes artifact existence, schema validity, dependency install, service reachability where specified.
- Agents log outcome (`READINESS_PASS`/`READINESS_FAIL`) with structured fields for each prerequisite.
- Registry map is refreshed (delta) prior to check to ensure latest artifact paths.
- Failures create `BLOCKER` issue and set task status to BLOCKED without consuming retry count.

Dependencies & Risks

- **False Negatives:** Superficial checks may pass but hidden issues later cause task failure; iterative improvement backlog.
- **False Positives:** Overly strict checks block progress; guideline for minimal viable verification needed.
- **Performance:** Large dependency graphs may slow checks; caching artifact metadata helps.
- **Distributed Variance:** Environment may change between check and execution; small race window tolerated.
- **Maintenance:** Checklist evolution needed as new artifact types introduced.

Summary

ADR-OS-013 enforces a lightweight but mandatory Readiness Check before executing any task, ensuring environmental prerequisites are satisfied, otherwise marking the task BLOCKED and raising a BLOCKER issue to trigger remediation.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What configurable timeout caps readiness check duration per task?	Hybrid_AI_OS	2025-06-27	OPEN	
2	Will there be centralized readiness check modules reusable across tasks?	Hybrid_AI_OS	2025-06-27	OPEN	
3	How are dynamic external dependencies (APIs) probed without causing side effects?	Hybrid_AI_OS	2025-06-27	OPEN	
4	Does a passed readiness check cache results for short window to avoid duplicate work?	Hybrid_AI_OS	2025-06-27	OPEN	
5	How are readiness assessments stored and linked to tasks for future audits?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix D lists readiness assessment schema proposal.
- Future: integrate service health endpoints into readiness evaluation.

Traceability

- adr_source: ADR-OS-013
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-014

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Guideline artifacts reside under `docs/guidelines/` and are referenced by artifact ID in registry.
- Agents always include relevant guideline context via `context_loading_instructions`; plan validator flags omissions.
- Guidelines are versioned with semantic tags; tasks pin a minimum compatible version.
- CI pipeline linter compares guideline checksum against registry to detect corruption.
- Guideline updates require CHANGELOG entry and reviewer approval label `guideline_update`.

Dependencies & Risks

- **Staleness:** Outdated guidelines risk incorrect behavior; quarterly review ceremonies scheduled.
- **Contradictions:** Multiple guideline docs may conflict; meta-guideline defines precedence rules.
- **Overload:** Excessive guidelines could bloat agent prompts; precision context loading mitigates.
- **Bypass:** Malicious plan might exclude guidelines; auto-validation blocks plan merge.
- **Maintenance Cost:** Continuous curation demands dedicated documentation steward role.

Summary

ADR-OS-014 establishes a version-controlled Project Guidelines store of Markdown artifacts that agents must load and follow, serving as a single source of truth for standards and preventing AI drift.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	How are deprecated guideline versions archived but still accessible for historical context?	Hybrid_AI_OS	2025-06-27	OPEN	
2	What tooling validates that every task's loaded guidelines cover required categories (e.g., testing, security)?	Hybrid_AI_OS	2025-06-27	OPEN	
3	Can guidelines embed executable checklists (YAML front-matter) for automated compliance validation?	Hybrid_AI_OS	2025-06-27	OPEN	
4	What is the conflict resolution policy when two guideline docs set differing conventions?	Hybrid_AI_OS	2025-06-27	OPEN	
5	Will there be a GUI editor with linting assistance to author guideline documents?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix G Frameworks Registry cross-references guideline categories.
- README doc section will link to guidelines index for quick navigation.
- Future feature: guideline recommendation engine based on task type.

Traceability

- adr_source: ADR-OS-014
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-015

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Pattern-based slicing supports regex compatible with Python `re` syntax.
- Orchestrator caches slice results keyed by artifact hash to avoid rereading unchanged files.
- Plan authors must include fallback logic (`load_full_if_pattern_missing: true/false`).
- Security: only whitelisted files may be sliced to avoid data leakage.
- Slicing overhead kept below 50ms per artifact on median hardware.

Dependencies & Risks

- **Pattern Drift:** Heading changes break patterns; plan validation tests headings existence.
- **Over-Slicing:** Too narrow slice omits needed context; can degrade task quality.
- **Performance:** Regex on large files may be slow; line number slicing preferred when possible.
- **Security Exposure:** Malicious pattern could read sensitive sections; file whitelist mitigates.
- **Debuggability:** Hard to trace what context was actually loaded; orchestrator logs slice metadata for audit.

Summary

ADR-OS-015 implements Precision Context Loading using optional `source_location_details` for line or pattern slicing, reducing token usage and context noise while maintaining relevant information for LLM agents.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	Will there be a visual diff tool to preview slice results during plan creation?	Hybrid_AI_OS	2025-06-27	OPEN	
2	How are binary or non-text artifacts handled—blocked or summarized automatically?	Hybrid_AI_OS	2025-06-27	OPEN	
3	Can multiple slice instructions target same artifact; how are they concatenated?	Hybrid_AI_OS	2025-06-27	OPEN	
4	What is the policy for specifying overlapping slices—deduplicate or preserve order?	Hybrid_AI_OS	2025-06-27	OPEN	
5	How does orchestrator flag pattern not found—BLOCKER issue or warning?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix D will include slice instruction schema examples.
- Future enhancement: semantic search slices via embedding similarity.

Traceability

- adr_source: ADR-OS-015
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-016

Initial Clarification Draft (TBD)

Assumptions & Constraints

- `exec_status_<g>.txt` writes are append-only events to minimize merge conflicts.
- A background StatusWriter agent batches updates every 2 seconds or 10 events, whichever first.
- Status file schema (`exec_status_schema.md`) includes vector_clock and trace_id per event.
- Maximum status file size before rotation: 5 MB; older events archived to `exec_status_<g>_archive_*.txt`.
- state.txt `current_exec_status_id_ref` is updated atomically with plan switch.

Dependencies & Risks

- **Write Contention:** High-frequency updates could still collide; batching + append-only mitigates.
- **Desync:** If state.txt points to wrong status file, dashboards misreport; health monitor validates linkage hourly.
- **Corruption:** Partial writes risk JSON breakage; writer uses temp + fsync and validation pass.
- **Scalability:** Many parallel plans may spawn numerous status files; indexing service required in future.
- **Observability Overhead:** Excessive event detail inflates status size; configurable verbosity levels.

Summary

ADR-OS-016 separates mutable execution telemetry from immutable plans via dedicated `exec_status_<g>.txt` files written in an append-only event stream, referenced from state.txt, allowing real-time progress tracking while preserving plan immutability.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What JSON event schema fields are mandatory for every status entry?	Hybrid_AI_OS	2025-06-27	OPEN	
2	How will multi-agent concurrent updates coordinate append offsets safely on network file systems?	Hybrid_AI_OS	2025-06-27	OPEN	
3	Will there be a gRPC or WebSocket stream to subscribe to status updates instead of file polling?	Hybrid_AI_OS	2025-06-27	OPEN	
4	What archival strategy compresses rotated status files to save space?	Hybrid_AI_OS	2025-06-27	OPEN	
5	How are status events reconciled after partition healing to ensure correct ordering?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix E Observability reports will summarize data pulled from status files.
- Potential enhancement: migrate status storage to an event log (e.g., Loki) with file gateway for backward compatibility.

Traceability

- adr_source: ADR-OS-016
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-017

Initial Clarification Draft (TBD)

Assumptions & Constraints

- MVP engine implemented in Python 3.11 leveraging Typer CLI and Pydantic for schema validation.
- Single-threaded sequential execution; concurrency left for Phase 2.
- Test project scaffold resides in `project_templates/`; engine assumes template availability.
- CI workflow executes engine on example plan to assert green build.
- MVP scope excludes LLM calls; any agent actions mocked via deterministic scripts.

Dependencies & Risks

- **Language Choice:** Python performance limits future scaling; rewrite risk.
- **Schema Volatility:** Early changes to schemas may break MVP; version pin and migration scripts needed.
- **Manual Plan Creation:** Human error in seed plan might skew MVP evaluation; validation tooling critical.
- **Limited Error Handling:** Happy-path focus could mask edge cases; follow-up issues expected.
- **Technical Debt:** Quick MVP decisions may require refactor in Phase 2; earmark debt items.

Summary

ADR-OS-017 kicks off Phase 1, delivering a command-line MVP engine capable of executing a single SCAFFOLDING plan end-to-end, proving the HAiOS file-based architecture and laying groundwork for future agent integration and concurrency.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|---------|---------|-----|-------|-----------------| | 1 | What Python packaging approach (poetry vs. pipenv) will the MVP adopt? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How will the MVP engine validate init and exec plan schemas before execution? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Which logging/telemetry library will capture distributed trace spans in the MVP? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | What automated tests define success for the MVP CI pipeline? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How will feedback from MVP runs feed into ADR or schema refinements for Phase 2? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|------------|------|-----------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix H CI policy includes job `mvp_engine_smoke_test`.
- Future: integrate LangChain-based agent wrappers once engine stable.

Traceability

- adr_source: ADR-OS-017
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-018

Initial Clarification Draft (TBD)

Assumptions & Constraints

- State snapshots are written every 60s or on graceful shutdown signal.
- Snapshot files stored under `os_root/snapshots/` using naming `snapshot_<g>.tar.zst`.
- Recovery process validates checksums and embedded version before replaying state.
- Secrets vault (`vault/secrets.json.gpg`) initialized with age-encryption; only supervisor key decrypts.
- Kill-switch flags monitored each event loop; emergency stop sets state.st to `BLOCK_INPUT`.

Dependencies & Risks

- **Snapshot Size:** Large workspaces may bloat snapshot archives; compression ratio tuning required.
- **Partial Snapshot:** Power loss mid-write could corrupt archive; write to temp then atomic rename.
- **Key Loss:** Losing supervisor private key renders vault unusable; offsite backup policy.
- **OPA Policy Drift:** Outbound whitelist may block legitimate new services until policy update.
- **Resource Limits:** Overly strict CPU/mem limits could kill long-running tasks; guideline for sizing.

Summary

ADR-OS-018 mandates periodic execution state snapshots, a secrets vault, kill-switch controls, resource limits, and outbound network policies to ensure recoverability and single-node zero-trust security foundation.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | What snapshot retention count or age policy prevents disk exhaustion? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are snapshot restores audited to prevent rollback attacks? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Can partial plan progress be resumed post-recovery without re-running completed tasks? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | What is the procedure for rotating secrets without downtime? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How will kill-switch events propagate to distributed agents in Phase 2? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix B contains incident response playbook referencing kill-switch flags.
- Appendix H CI step `snapshot_integrity` verifies archive round-trip restore.

Traceability

- adr_source: ADR-OS-018
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-019

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Prometheus endpoint bound to 127.0.0.1 unless `haios.config.json.observability.public` set true.
- Metric cardinality guidelines restrict label combinations to <10k series.
- CostMeter converts OpenAI tokens to USD via configurable `usd_per_token` fetched daily.
- Grafana dashboards stored in JSON; editing via UI exports committed by CI helper script.
- Budget thresholds configurable per environment (dev/stage/prod) via config overlay.

Dependencies & Risks

- **Metric Explosion:** Uncontrolled labels could overload Prometheus; enforced regex lint.
- **Token Cost Drift:** API price changes may skew budgets; daily cron updates exchange rate.
- **Port Exposure:** /metrics endpoint may leak info; reverse proxy restricts.
- **Alert Fatigue:** Too many budget alerts reduce signal; thresholds tuned via SLO review.
- **Storage Retention:** Long-term traces consume disk; Loki/tempo integration backlog.

Summary

ADR-OS-019 introduces a unified Observability & Budget Framework using Prometheus metrics, OpenTelemetry traces, Grafana dashboards, and CostMeter budget enforcement to provide real-time visibility and guard-rails for resource usage.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|---------|---------|-----|-------|-----------------| | 1 | What default retention period will Prometheus use for metrics in dev vs prod? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are secrets (e.g., API keys) scrubbed from traces and logs before export? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Will budget enforcement support per-task overrides for resource-intensive migrations? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | Which alert channels (Slack, email) are integrated for budget and health alerts? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How is metric taxonomy versioned to avoid breaking dashboards when labels change? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|------------|-----|-----------|------------------|--------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix H Observability policy references control IDs 19-O1■O7.
- Grafana dashboard JSON validated by `grafana-dashboard-linter` in CI.

Traceability

- adr_source: ADR-OS-019
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-020

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Default runtime.mode is STRICT; CLI `--mode` override respected only in non-CI environments.
- DEV_FAST artifacts carry `"dev_mode": true` flag in registry and filename suffix `_devfast`.
- Validators block mixing modes: STRICT plan cannot depend on devfast artifacts.
- Mode setting is logged at engine startup with g event and trace id for audit.
- CI job enforces STRICT via environment variable `HAIOS_FORCE_MODE=STRICT`.

Dependencies & Risks

- **Mode Drift:** Developers may forget to switch back to STRICT before commit; pre-commit hook warns.
- **Artifact Pollution:** Devfast files may leak into repository; CI fails if detected.
- **Security Loophole:** Devfast skips some checks; misuse in shared env risky; enforcement via config.
- **Complex Config:** Future additional modes could complicate matrix; document contribution guidelines.
- **User Confusion:** Clear docs and CLI help required to explain behaviours.

Summary

ADR-OS-020 defines two runtime modes—STRICT and DEV_FAST—enabling developers to trade safety for speed locally while preserving rigorous policy in CI and production through artifact labelling and validator enforcement.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | How does pre-commit hook detect leftover devfast artifacts before push? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | Will the engine refuse to run in DEV_FAST if budgets are undefined? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Can individual tasks request STRICT checks even in DEV_FAST plan (e.g., snapshot)? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | What visual indicator will cockpit UI display for mode (e.g., banner colour)? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | Is telemetry reduced in DEV_FAST to lower overhead, and how is that flagged in traces? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix H CI policy step `devfast_artifact_scan` enforces pollution guard.
- README update will include mode explanation and developer workflow examples.

Traceability

- adr_source: ADR-OS-020
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-021

Initial Clarification Draft (TBD)

Assumptions & Constraints

- All new ADRs, plans, and connectors must include explicit Assumptions, Confidence, Self-Critique, and Clarifying Questions sections.
- CI linter (`assumption_lint`) enforces non-empty lists; placeholder text fails build.
- Confidence levels limited to enum {High, Medium, Low} for consistency.
- Legacy artifacts are scheduled for retrofit within two release cycles; technical-debt label tracks progress.
- Templates reside in `docs/templates/adr_template.md` and are version-locked via checksum.

Dependencies & Risks

- **Author Burden:** Added sections may slow documentation; training and examples mitigate.
- **Superficial Compliance:** Writers may add low-value boilerplate; peer review and CI heuristic checks needed.
- **Template Drift:** Changes require mass update of artifacts; automated script maintained.
- **Linter False Positives:** Strict parsing may block merges; incremental rollout with warning mode first.
- **Cultural Resistance:** Teams unused to self-critique may push back; governance board champions adoption.

Summary

ADR-OS-021 mandates explicit assumption surfacing with confidence and self-critique across all system artifacts, enforced by templates and CI linting to reduce hidden risks and improve traceability.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|---------|---------|-----|-------|-----------------| | 1 | What heuristic will CI use to detect low-value placeholder text in assumption lists? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are confidence levels aggregated for overall artifact risk scoring? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Is a migration script provided to backfill assumption sections in legacy ADRs? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | Can exceptions be granted for small utility scripts, and how documented? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How will periodic re-audit of assumptions be scheduled and tracked? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|------------|-----|-----------|------------------|--------| | 1 | *placeholder* | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix E reporting guidelines reference assumption metrics for quality dashboards.
- Future: integrate automatic sentiment analysis to flag superficial self-critique.

Traceability

- adr_source: ADR-OS-021
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-022

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Inventory arrays embedded in annotation blocks must not exceed 1 000 items; GC trims older EXPIRED entries.
- Delta logs are append-only; size capped at 2 MB before rotation to `inventory_delta_<g>_arch1.log`.
- Janitor cleanup cadence default 100 global events, configurable via `haios.config.json.inventory.gc_interval_g`.
- Reservation TTL default 30 minutes if `ttl_seconds` absent.
- Builder agents have read-only access; any write attempt raises `PERMISSION_DENIED` Issue.

Dependencies & Risks

- **Zombie Reservations:** Agents crash after RESERVE; janitor must roll back; monitor orphan rate.
- **Merge Conflicts:** Manual edits to annotation may conflict with delta replay; automated patcher preferred.
- **Bloat:** Large code snippets inflate annotation; guideline recommends storing pointer to file instead.
- **Concurrency:** Simultaneous RESERVE events could race; optimistic g ordering resolves but edge cases logged.
- **Schema Evolution:** Adding new lifecycle states requires migrator script and CI schema bump.

Summary

ADR-OS-022 introduces a two-tier Mechanical Inventory Buffer stored in annotation blocks with append-only delta logs, enabling agents to reuse resources, prevent redundant work, and maintain auditability.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | What tooling visualizes current inventory and reservation status for supervisors? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are large binary artifacts referenced—stored externally with hash pointer or base64 in inventory? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Can inventory items be shared across initiatives, and how is namespace collision avoided? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | What audit metrics track inventory churn and janitor rollbacks? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How does delta log replay handle out-of-order events after partition healing? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix D schema updates include inventory v2.1 fields.
- Future: integrate UI to manually expire or promote inventory items.

Traceability

- adr_source: ADR-OS-022
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-023

Initial Clarification Draft (TBD)

Assumptions & Constraints

- All mutable API endpoints require `Idempotency-Key` header; keys UUIDv7 with 24h TTL.
- Client libraries provide retry wrapper implementing exponential backoff (base 1s, factor 2, jitter 0-0.5).
- Circuit breaker trips after 5 consecutive failures or 30s rolling error rate >50%.
- Idempotency key store uses os_root/kv/idempotency_.sqlite for Phase-1.
- GET/HEAD requests exempt but still include trace_id for linkage.

Dependencies & Risks

- **Key Store Size:** High throughput may bloat SQLite; pruning job deletes keys past TTL.
- **Replay Attacks:** Attacker reuses key; include HMAC of user token in key or signed header.
- **Complex Integration:** External APIs lacking idempotency require adapter or compensation logic.
- **Configuration Drift:** Services may deviate from standard parameters; linter tests contract.
- **Latency:** Extra lookup adds ms latency; acceptable for reliability gains.

Summary

ADR-OS-023 enforces universal idempotency using Idempotency-Key headers and standardized exponential-backoff retry with circuit breakers, ensuring safe operation and preventing duplicate side-effects across all OS communications.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | Should idempotency keys be namespaced per endpoint or global? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How will key storage migrate from SQLite to distributed store in Phase-2? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Can read-modify-write operations use same key for GET and subsequent POST? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | What metrics monitor retry attempts and circuit breaker trips? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How are long-running saga steps made idempotent across partial failures? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix H CI test `idempotency_enforcement` checks mandatory header on integration tests.
- CostMeter records additional CPU cost for retries for budget tracking.

Traceability

- adr_source: ADR-OS-023
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-024

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Primary message bus for Phase-1: NATS JetStream running locally via Docker compose.
- Event schema uses CloudEvents v1.0 with mandatory fields: id, source, type, specversion, time, trace_id.
- Saga coordinator agent persists state in `os_root/sagas/` with idempotent updates.
- Dead-letter queue retention 7 days; monitored by alert.
- Consumers must ack within 30s or message redelivered with exponential delay.

Dependencies & Risks

- **Bus Downtime:** Single-node NATS may fail; dev environment acceptable, prod requires cluster.
- **Schema Evolution:** Event version bumps may break consumers; adopt versioned type names.
- **Message Loss:** JetStream persistence mitigates; still need periodic snapshot.
- **Debug Complexity:** Asynchronicity complicates trace; OpenTelemetry baggage carries trace_id.
- **Out-of-Order:** Consumers design for idempotent handling; event numbering optional.

Summary

ADR-OS-024 establishes standard asynchronous communication patterns using NATS JetStream, CloudEvents schemas, and saga coordination to achieve eventual consistency and decoupled workflows across HAiOS agents.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What naming convention prefixes event `type` field to indicate domain?	Hybrid_AI_OS	2025-06-27	OPEN	
2	How will multi-tenant message bus namespaces be organized in future phases?	Hybrid_AI_OS	2025-06-27	OPEN	
3	Should saga state be stored in exec_status files or separate store?	Hybrid_AI_OS	2025-06-27	OPEN	
4	What tooling visualizes saga progress for supervisors?	Hybrid_AI_OS	2025-06-27	OPEN	
5	How are compensating actions audited and linked to original events?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix G Frameworks Registry will include event naming standard reference.
- Future: evaluate Kafka for Phase-2 distributed deployment.

Traceability

- adr_source: ADR-OS-024
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-025

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Internal requests authenticated via short-lived PASETO tokens (TTL 5 min) issued by an in-process AuthService.
- Mutual TLS certificates issued by local CA rotated every 24 h by Supervisor automation.
- Service ACLs defined in `os_root/security/policies/rbac.yaml` and enforced by side-car middleware.
- Bootstrap secrets delivered via age-encrypted file unlocked by operator during install.
- Token validation cache (LRU, 1 k entries) keeps latency <2 ms per call.

Dependencies & Risks

- **PKI Complexity:** Certificate issuance failures may block service start; fallback self-signed dev mode.
- **Token Replay:** Clock skew could allow reuse; include nonce + store last 100 nonces per service.
- **Performance:** mTLS handshake adds latency; enable HTTP/2 keep-alive.
- **Secret Leakage:** Misconfigured logging may print tokens; log filter middleware strips Authorization headers.
- **Revocation:** Compromised tokens require immediate invalidation; implement in-memory deny list broadcast via NATS.

Summary

ADR-OS-025 enforces zero-trust by requiring PASETO tokens, mutual TLS, and least-privilege RBAC for all internal calls, eliminating implicit trust and shrinking blast radius.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|---------|---------|-----|-------|-----------------| | 1 | Which PASETO version (v2 local/public) will be standard? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are token signing keys rotated and distributed securely? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Will service meshes (e.g., Linkerd) replace custom mTLS in Phase-2? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | How does RBAC policy file get validated and loaded at runtime? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | What monitoring alerts fire on repeated auth failures indicating attack? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|------------|-----|-----------|------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix H security policy references zero-trust enforcement scripts.
- Future: integrate SPIFFE/SPIRE for automated workload identity.

Traceability

- adr_source: ADR-OS-025
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-026

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Registry heartbeats every 15 s; missing 3 heartbeats → UNHEALTHY.
- /health endpoint returns JSON {status, version, g, uptime_seconds} with 200 OK.
- Status stream delivered via NATS subject `topology.status` (JetStream durable).
- Registry HA through raft–based leader election; at least 3 replicas in prod.
- Subscribers cache last status per service for 60 s fallback.

Dependencies & Risks

- **Registry Outage:** Loss of quorum halts registration; fallback read-only mode using last snapshot.
- **Heartbeat Flood:** Thousands of agents may overload; adaptive backoff supported.
- **False Positives:** Short GC pauses might miss heartbeat; tolerate one skip before mark degraded.
- **Security:** Status stream messages signed to prevent spoofing.
- **Config Drift:** Different heartbeat intervals cause noisy health; central config enforced.

Summary

ADR-OS-026 defines dynamic topology management through a HA agent registry, standardized /health checks, periodic heartbeats, and a push-based status stream, enabling rapid failure detection and adaptive scaling.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|---------|---------|------|--------|-----------------| | 1 | What JSON schema version governs /health response fields? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are registry replicas discovered and bootstrap elected? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Will degraded services still receive traffic or be shadowed? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | How is heartbeat interval negotiated during high load? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | What dashboard visualizes overall system topology and health? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix E includes Prometheus rules `haios_service_up` utilising /health scrape.
- Future: explore gossip-based SWIM protocol to replace central polling.

Traceability

- adr_source: ADR-OS-026
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-027

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Vector clock field `vc` added to all status events and delta logs.
- g counter provides total ordering within single node; between nodes vector clocks resolve causality.
- Clock skew tolerated up to 1 s for trace timestamps; NTP sync recommended.
- Merge strategy: when two divergent event logs detected, reconciliation task merges by vc dominance.
- Tooling `vc_merge.py` shipped in scripts/ for conflict resolution.

Dependencies & Risks

- **Vector Size:** Many nodes enlarge vc; compression algorithm delta-encodes zeros.
- **Complex Reconciliation:** Manual intervention may be needed on three-way conflict.
- **Performance:** vc comparison adds cpu overhead; negligible for <50 nodes.
- **Skew:** Unsynced clocks affect human-readable time but not vc causality.
- **Data Loss:** Log truncation could lose vc history; periodic snapshot includes last vc.

Summary

ADR-OS-027 mandates logical vector clocks attached to every state-changing event, supplementing the monotonic g counter to ensure causal ordering across distributed nodes.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | What max node count is supported before vc becomes unwieldy? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are vc fields represented in JSON—array or map of node id to g? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Will reconciliation be automated or require human approval? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | How does vc integrate with OpenTelemetry span relationships? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | What testing ensures vc merge logic handles all edge cases? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix E provides algorithm pseudocode for vc compare & merge.

Traceability

- adr_source: ADR-OS-027
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-028

Initial Clarification Draft (TBD)

Assumptions & Constraints

- state.partition_status field enumerates CONSISTENT | PARTITIONED | RECONCILING.
- Minority partition enters read-only mode; mutating operations raise PARTITION_ERROR.
- Healing detected via successful gossip ping 3x; RECONCILING tasks merge vc and g counters.
- Snapshot taken pre-reconcile to enable rollback if merge fails.
- Partition detection uses heartbeat absence >45 s.

Dependencies & Risks

- **Split Brain:** Concurrent writes in partitions risk divergence; read-only enforcement critical.
- **Delayed Detection:** Long heartbeat interval delays partition awareness; tune per deployment.
- **Merge Conflicts:** Irreconcilable changes may require manual resolution; create ISSUE partition_conflict.
- **Performance:** Gossip pings add network chatter; batch round trips.
- **Testing:** Simulating partitions in CI requires network emulation.

Summary

ADR-OS-028 defines partition tolerance strategy: detect network partitions via heartbeat, switch minority partitions to read-only, and reconcile state using vector clocks and snapshots upon healing to maintain consistency.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | What automated tests verify read-only enforcement during partition? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How is g counter advanced in minority partition without writes? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | What UI signal alerts operators to PARTITIONED state? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | Can tasks be explicitly marked partition-tolerant to continue? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How long will RECONCILING mode last before forced human intervention? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix B incident response includes partition healing runbook.

Traceability

- adr_source: ADR-OS-028
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-029

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Every function boundary propagates trace_id via OpenTelemetry context vars.
- Root span name pattern: `haios.<phase>.<plan_id>`.
- Traces exported to OTLP endpoint `http://localhost:4318` in dev; configurable.
- Sampling rate 100% in STRICT, 10% in DEV_FAST unless error spans.
- Logs include trace_id in structured JSON field for correlation.

Dependencies & Risks

- **High Volume:** 100% sampling may overwhelm collector; adaptive sampling roadmap.
- **Missing Context:** Legacy code paths may drop context; tracer lint scans import graph.
- **Sensitive Data:** PII must be scrubbed from span attributes; allowlist enforced.
- **Exporter Failure:** If OTLP unavailable, spans buffered to disk up to 100 MB.
- **Overhead:** Tracing adds ~5 µs per span; acceptable.

Summary

ADR-OS-029 mandates universal trace propagation through OpenTelemetry, ensuring every event, metric, and log carries a trace_id for end-to-end observability and debugging.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	Will span batching be enabled to reduce network overhead?	Hybrid_AI_OS	2025-06-27	OPEN	
2	How are trace IDs persisted in snapshots to resume correlation after restart?	Hybrid_AI_OS	2025-06-27	OPEN	
3	What tools visualize traces for non-HTTP internal events?	Hybrid_AI_OS	2025-06-27	OPEN	
4	How are sensitive attributes redacted before export?	Hybrid_AI_OS	2025-06-27	OPEN	
5	What SLA defines acceptable trace export lag?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix H observability defines alert when trace error ratio >3%.

Traceability

- adr_source: ADR-OS-029
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-030

Initial Clarification Draft (TBD)

Assumptions & Constraints

- Models registry file `docs/frameworks_registry.md` lists canonical models with id, version, compliance tests.
- Agents reference models via `/registry/{id}/{version}` to avoid ambiguity.
- CI job validates each ADR cites model ids existing in registry.
- Deprecated models flagged `status: DEPRECATED`; use triggers linter warning.
- Registry updates require semantic version bump and changelog entry.

Dependencies & Risks

- **Registry Drift:** Failure to update registry breaks compliance links; governance review process.
- **Version Proliferation:** Many versions clutter; adopt LTS policy.
- **Broken Links:** ADR references to removed models flagged in CI.
- **Human Error:** Manual editing mistakes; JSON schema validation for registry file.
- **Adoption Lag:** Agents may not upgrade to new model guidelines quickly; compatibility matrix maintained.

Summary

ADR-OS-030 defines a canonical models & frameworks registry, ensuring all architectural documents reference standardized principles and enabling automated compliance checks.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | What metadata fields are mandatory for each registry entry? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | How are breaking changes communicated to downstream teams? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | Will a REST endpoint mirror the file for runtime lookups? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | What tool auto-generates compliance badges for ADR headers? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | How does registry handle forks/extensions of existing models? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Registry schema documented in Appendix G.

Traceability

- adr_source: ADR-OS-030
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-031

Initial Clarification Draft (TBD)

Assumptions & Constraints

- ADR-OS-031 presumably short; placeholder logic: treat as minor feature tag? (since content unknown)
- Pre-initiative artifacts must be agent-parseable Markdown, not binary formats.
- CI linter `preinit_artifact_lint` verifies presence of Vision, PRD/MRD, TRD, Assumption Register, Diagrams, and Execution Outline.
- Diagram files stored in `docs/diagrams/` as Mermaid or SVG; each referenced via relative link.
- Minimal fast-path template allowed only for initiatives tagged `complexity: trivial` and still requires Assumption register.
- Artifact versions tracked via front-matter `artifact_version`; changes trigger compliance re-check.

Dependencies & Risks

- **Author Overhead:** Comprehensive artifact set may slow kickoff; templating tools mitigate.
- **Staleness:** Artifacts may drift during long initiatives; quarterly review schedule enforced.
- **Diagram Quality:** Poor diagrams reduce clarity; CI uses Mermaid syntax validation.
- **Compliance Lint False Positives:** Strict checks may block merge; allow override with `needs_override` label after review.
- **Version Conflicts:** Multiple versions of artifacts may confuse agents; registry stores only latest plus archive.

Summary

ADR-OS-031 mandates a rigorous set of standardized, agent-readable pre-initiative source documents (Vision, PRD/MRD, TRD, Assumption Register, Diagrams, Execution Outline) to eliminate ambiguity and context drift before planning begins.

Clarification Questions

#	Question	Asked By	Date	Status	Response Summary
1	What automation will generate skeleton pre-initiative artifact templates?	Hybrid_AI_OS	2025-06-27	OPEN	
2	How are small spikes exempted or streamlined while maintaining rigor?	Hybrid_AI_OS	2025-06-27	OPEN	
3	Will diagram compliance accept PlantUML in addition to Mermaid?	Hybrid_AI_OS	2025-06-27	OPEN	
4	How is artifact review/approval workflow integrated with GitHub PRs?	Hybrid_AI_OS	2025-06-27	OPEN	
5	What metrics track artifact freshness and completeness across portfolio?	Hybrid_AI_OS	2025-06-27	OPEN	

Responses

#	Response By	Date	Related Q#	Related Dissent #	Summary
1	*placeholder*				

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix C provides template Markdown files for each required artifact type.
- Future enhancement: VS Code extension to validate pre-initiative documents in real time.

Traceability

- adr_source: ADR-OS-031
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [ ] Idempotent updates supported
- [ ] Message-driven integration points documented
- [ ] Immutable audit-trail hooks attached

ADR Clarification Record: ADR-OS-032

Initial Clarification Draft (TBD)

+This clarification aligns with ADR-OS-032 (Canonical Models and Frameworks Registry & Enforcement) to surface assumptions, track open questions, and ensure downstream artifacts conform to the registry mandate.

Assumptions & Constraints

- Best-practice models/frameworks must be explicitly cataloged and referenced, not implicit.
- Registry is versioned, agent-readable (YAML or JSON front-matter), stored at `docs/frameworks_registry.md`.
- CI job `framework_compliance_check` blocks merges if required frameworks are missing or compliance unproven.
- Governance board owns registry updates; changes require semantic version bump.
- Enforcement mode tiers (Required, Recommended, Optional) map to severity levels in lint output.

Dependencies & Risks

- **Registry Drift:** Without ownership and review cadence, models may become outdated.
- **Over-Enforcement:** Rigid checks could stifle experimentation; override process needed.
- **Complex Onboarding:** New contributors must learn registry semantics; documentation/training essential.
- **Tooling Maintenance:** Linter and VS Code plugin must evolve with schema changes.
- **Cross-ADR Alignment:** Must stay synced with ADR-OS-021 (runtime metadata) for enforcement hooks.

Summary

ADR-OS-032 establishes a central, versioned registry of canonical models and frameworks and mandates that every major artifact declare compliance, exceptions, and proof against that registry. It enables automated enforcement via CI/lint and provides legible standards for humans and agents.

Clarification Questions

| # | Question | Asked By | Date | Status | Response Summary | |---|----------|----------|------|--------|------------------| | 1 | Which frameworks are **system mandatory** vs **recommended** at launch? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 2 | What governance process approves new registry entries or deprecations? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 3 | How will exceptions be tracked (e.g., annotation vs issue label)? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 4 | Where does the linter output compliance proof artifacts (reports)? | Hybrid_AI_OS | 2025-06-27 | OPEN | | | 5 | Can registry entries reference external standards bodies (e.g., ISO) by link? | Hybrid_AI_OS | 2025-06-27 | OPEN | |

Responses

| # | Response By | Date | Related Q# | Related Dissent # | Summary | |---|-------------|------|------------|-------------------|---------| | 1 | *placeholder* | | | | |

Formal Reviews & Dissents

<-- help test -->

Additional Notes

- Appendix G (`Frameworks_Registry`) details current registry content.
- Long-term plan: GraphQL endpoint exposing registry for IDE plugins.

Traceability

- adr_source: ADR-OS-032
- trace_id: {{TRACE-ID}}
- vector_clock: {{VECTOR-CLOCK}}

Distributed-Systems Protocol Compliance Checklist

- [x] Idempotent updates supported (registry updates are version-controlled)
- [x] Message-driven integration points documented (CI linter emits events)
- [x] Immutable audit-trail hooks attached (Git history + signed commits)

ADR Clarification Records – Index (g70)

| ADR | Title | File | |-----|-------|------| | ADR-OS-001 | Clarification of five-phase operational loop & state management | ADR-OS-001_clarification.md | | ADR-OS-002 | Hierarchical planning store lineage guarantees | ADR-OS-002_clarification.md | | ADR-OS-003 | EmbeddedAnnotationBlock embedding strategy | ADR-OS-003_clarification.md | | ADR-OS-004 | Global event & version counters concurrency model | ADR-OS-004_clarification.md | | ADR-OS-005 | Configuration-driven directory structure | ADR-OS-005_clarification.md | | ADR-OS-006 | Schema-based scaffolding system | ADR-OS-006_clarification.md | | ADR-OS-007 | Evidence-based testing lifecycle | ADR-OS-007_clarification.md | | ADR-OS-008 | Narrative Markdown reports generation | ADR-OS-008_clarification.md |

_Last updated: global event counter **g=70** via `plan_adr_clarification_cleanup_g66` (task `cross_link_registry`).