Aurora Workflow Orchestration – Method Specification v1.2.1

Shawn C. Wright, Aurora Research Initiative / Waveframe Labs Division

October 2025

Aurora Workflow Orchestration (AWO)

Method Specification — v1.2.1 (Scaffold)

Maintainer: Waveframe Labs

Preface

This document defines the **normative specification** for Aurora Workflow Orchestration (AWO).
It replaces descriptive or philosophical language with enforceable procedural logic.
All future automation layers (e.g., CRI-CORE) must validate conformance against these requirements.

Interpretation of Compliance Language
- **MUST** — absolute requirement for AWO-compliant repositories.
- **SHOULD** — strong recommendation; deviations must be justified in documentation.
- **MAY** — optional behavior permitted for flexibility.

---

1. Introduction

1.1 Purpose

Aurora Workflow Orchestration (AWO) establishes a formal, falsifiable framework for conducting reproducible AI-assisted research.
It defines the structural and procedural rules by which reasoning processes—whether human, synthetic, or hybrid—are documented, attested, and version-controlled.

This specification is **methodological**, not philosophical.
It governs the organization, validation, and archival of reasoning artifacts so that every claim produced under AWO can be independently verified.

1.2 Scope

This document applies to all research workflows that:

- Integrate AI or automated reasoning systems as active participants in the research process.

- Produce verifiable artifacts such as manifests, runs, and audit logs.

- Intend for those artifacts to be **reproducible, falsifiable, and citable**.

It defines the **minimum structural and procedural requirements** for an AWO-compliant repository, including file hierarchy, provenance recording, versioning, and attestation rules.

AWO does **not** specify runtime behavior or enforcement mechanisms.
Those are defined in successor frameworks such as **CRI-CORE**, which must implement this specification as their normative foundation.

1.3 Objectives

The objectives of the AWO standard are to:

1. Encode the **scientific method** as a verifiable workflow rather than a descriptive ideal.

2. Replace subjective credibility with **objective auditability**.

3. Ensure that every reasoning artifact—data, model, or decision—can be traced to its origin.

4. Provide a foundation for automated reproducibility enforcement systems.

5. Support both manual and fully automated orchestration without altering compliance semantics.

1.4 Relationship to Other Documents

- The **AWO Whitepaper** provides conceptual background and philosophical rationale.

- The **AWO Adoption Guide** describes practical implementation and onboarding.
- This **Method Specification** defines the normative requirements that all AWO artifacts must satisfy.

Where discrepancies occur, **this specification takes precedence**.

1.5 Normative References

- **AWO Whitepaper v1.1** (Waveframe Labs)
- **Aurora Workflow Orchestration Adoption Guide v1.2.1**
- **Architecture Decision Records (ADR-0001 – ADR-0017)** — authoritative design decisions underlying AWO's structural, governance, and lifecycle model.
- **CRI-CORE Design Notes** (draft, forthcoming)
- **ISO/IEC Directives Part 2** — interpretation of compliance terms ("shall," "should," "may")

1.6 Status of This Version

Version 1.2.1 represents the **finalized methodological form** of AWO under Waveframe Labs governance.
Future revisions may clarify or extend definitions for CRI-CORE compatibility but will not alter the normative logic without an explicit version increment.

---

2. Definitions

This section defines the key entities and concepts used throughout the Aurora Workflow Orchestration (AWO) standard.
All terms are **normative** unless otherwise specified.

Wherever applicable, definitions align with terminology used in the AWO Whitepaper and will later be cross-referenced to CRI-CORE schema identifiers.

2.1 Core Entities

Run
A discrete, traceable research execution instance.
Each Run represents a bounded reasoning process that produces one or more verifiable artifacts and is identified by a unique timestamp or run ID.
All Runs must be immutable once attested.

Provenance
The complete, chronological lineage of data, logic, parameters, and decisions leading to a result.
Provenance includes all intermediate steps, transformations, and validations necessary to reproduce a Run.
In CRI-CORE, this concept maps to the `provenance-ledger` schema.

Artifact
Any persistent output generated within an AWO process.
Artifacts include reports, manifests, ADRs, checksums, datasets, logs, or schema validation results.
Artifacts must be versioned, hashable, and linkable to a Run.

Attestation
A confirmation—human, automated, or hybrid—that artifacts produced during a Run are complete, correct, and verified against defined falsifiability criteria.
Attestations are recorded in `approval.json` and form the evidentiary basis for repository integrity.

ADR (Architecture Decision Record)
A structured document that records a significant reasoning or design choice, the context in which it was made, and its consequences.
ADRs form the canonical log of epistemic evolution and are numbered sequentially (`ADR-0001` to `ADR-NNNN`).
Each Run must reference at least one ADR.

Manifest (Falsifiability Manifest)
A declaration of the hypothesis, predicted outcomes, and explicit disproof conditions for a Run.
The Manifest defines what constitutes falsification before execution.
It serves as the precondition for attestation and must be stored under `/docs/`.

2.2 Secondary Concepts

Repository
The complete version-controlled environment in which all AWO artifacts are stored.
Every AWO-compliant Repository must maintain a standard directory structure defined in Section 4.

Role
A functional agent—human or synthetic—responsible for a specific epistemic operation within the reasoning lifecycle (see Section 3).
Roles are procedural, not hierarchical.

Conformance
The degree to which an AWO repository satisfies all mandatory requirements defined in this specification.
Conformance is binary (pass/fail) for each clause but may include graded compliance levels ("Minimum," "Standard," "Full") as defined in the Adoption Guide.

Attestation Record
The recorded output of a completed review or validation step, stored as a structured file (`approval.json`) under the corresponding Run directory.
It includes participant identity, timestamp, and signature or digital hash.

2.3 Future Schema Alignment

All defined entities in this section will be mapped to corresponding CRI-CORE schema classes in later versions.
Cross-references will be introduced once the enforcement layer is finalized.

TODO: Refine definitions list and cross-link to CRI-CORE schema references after CRI draft publication.

---

3. Roles and Responsibilities

3.1 Overview

AWO defines **roles** as functional agents within a reasoning workflow, not as human job titles.
Each role represents a discrete epistemic operation necessary to ensure falsifiability, reproducibility, and integrity.
Roles may be embodied by humans, AI systems, or hybrid arrangements, but the **responsibility and accountability structure** must remain explicit and verifiable.

Every AWO-compliant Run MUST declare the roles involved and the corresponding participants (human or synthetic).
Multiple roles MAY be fulfilled by a single agent if traceability and attestation integrity are preserved.

3.2 Canonical Roles

Role	Core Function	Description	Typical Implementation
Orchestrator	Governance and context management	Governs execution order, maintains reasoning context, and determines when to fork, merge, or conclude runs. Responsible for continuity, documentation, and decision routing.	Human-in-the-loop controller or primary model agent (e.g., lead researcher, workflow coordinator).
Voter / Evaluator	Comparative validation	Compares multiple reasoning paths or outputs, ranks them by internal consistency and falsifiability, and selects the candidate most aligned with predefined criteria.	Model ensemble, peer review, or statistical evaluator.
Auditor	Verification and compliance	Independently checks reasoning validity, schema adherence, falsifiability, and traceability to prior evidence. Approves or rejects attestation claims.	Dedicated verification model, CI validator, or human reviewer.

Role	Core Function	Description	Typical Implementation
Synthesizer / Consensus	Result consolidation	Merges validated reasoning threads into a coherent, singular artifact. Produces the final report or output from attested inputs.	Aggregation model, summarizer, or post-processing layer.
Critic / Red Team (optional)	Adversarial robustness testing	Generates counterarguments or adversarial reasoning challenges to expose weaknesses in claims before attestation.	Adversarial model, external reviewer, or dedicated counterfactual analysis agent.

3.3 Role Interactions

- **Sequential Integrity:** Roles SHOULD execute in a reproducible order—Orchestrator → Evaluator → Auditor → Synthesizer.
  Optional Critic roles MAY interject between Evaluator and Synthesizer stages.
- **Non-Circular Validation:** The same agent MUST NOT serve as both Orchestrator and Auditor within the same Run unless explicitly justified and recorded in the attestation log.
- **Attestation Requirements:** Each Run MUST include a record of which roles were fulfilled, by whom, and under what authority.
- **Traceability Obligation:** Artifacts and logs MUST explicitly reference the roles responsible for their generation or validation.

3.4 Role Attribution and Record-Keeping

- Every `approval.json` file MUST list all participating roles and their associated agent identifiers (human name, model ID, or process hash).

- When multiple roles are automated, their decision boundaries MUST be defined in the workflow manifest or configuration file.
- Manual overrides or deviations from standard AWO behavior MUST be documented under /logs/overrides/ and cross-referenced to the applicable ADR

3.5 Compliance and Auditing

- AWO-compliant repositories MUST demonstrate separation of governance (Orchestrator) and verification (Auditor).
- Each role's output MUST be traceable to an ADR or manifest entry.
- Automated systems fulfilling these roles MUST log model versions, prompt contexts, and decision justifications to ensure reproducibility.
- Failure to document role interactions constitutes a **non-conformance** condition under this specification.

3.6 Future Role Extensions

Future AWO or CRI-CORE revisions MAY extend the canonical role set (e.g., **Planner**, **Historian**, **Meta-Auditor**) as automated reasoning matures.
Any such extensions MUST maintain backward compatibility with this role schema and preserve attestation semantics.

TODO: Cross-link these roles to CRI-CORE validation modules (e.g., `orchestrator-agent`, `auditor-module`, `consensus-engine`) once defined.

---

4. Repository Requirements

4.1 Purpose and Scope

This section defines the mandatory and recommended structural requirements for an AWO-compliant repository.
These requirements ensure that every research artifact is **traceable**, **auditable**, and **reproducible** without external dependencies.
All provisions in this section are **normative** unless explicitly labeled "informative."

4.2 Core Directory Structure

An AWO-compliant repository **MUST** include the following top-level directories:

Directory	Purpose	Requirement
`/docs/`	Contains all formal documents (Whitepaper, Method Spec, Adoption Guide, PDFs, and audit summaries).	MUST
`/decisions/`	Contains all Architecture Decision Records (ADRs). Each ADR MUST be timestamped and sequentially numbered.	MUST
`/logs/`	Houses all workflow, audit, and override logs (see ADR-0004).	MUST
`/schemas/`	Stores validation schemas and structure definitions for manifests, runs, and audits.	SHOULD
`/templates/`	Contains boilerplate forms for manifests, audit reports, and ADRs.	SHOULD
`/runs/`	Contains attested execution outputs, including manifests, reports, approvals, and checksums.	MUST
`/figures/`	Contains diagrams, charts, and other non-textual documentation.	SHOULD

Directory	Purpose	Requirement
`/workflows/`	Contains procedural examples or reproducible automation steps (optional, pre-CRI).	MAY

The repository root **MUST** contain: - `README.md` — entry point and index. - `CHANGELOG.md` — lifecycle record of repository evolution. - `SHA256SUMS.txt` — integrity registry for all signed artifacts. - `LICENSE` and `LICENSE-CC-BY-4.0.md` — primary and documentation licenses. - `.github/workflows/` — automated build and PDF pipelines.

4.3 Log Directory Specification

Each AWO-compliant repository **MUST** implement the following substructure within `/logs/`:

Subfolder	Description	Reference
`/logs/workflow/`	Chronological records of human and agent activity, covering decisions, forks, merges, and context.	ADR-0004
`/logs/audits/`	Independent audit results, rejection events, or revalidation findings.	ADR-0003
`/logs/overrides/`	Manual interventions, rationale, and signatures for non-automated overrides.	ADR-0004, ADR-0012

Log entries **MUST** follow the schema outlined in ADR-0004, including timestamps, participant IDs, impacted artifacts, and outcome codes.
Each log file **MUST NOT** be modified retroactively after attestation.

4.4 ADR Requirements

- ADRs **MUST** follow sequential numbering (`ADR-0001` through `ADR-NNNN`) and reside in `/decisions/`.

- Each ADR **MUST** contain:
  - Title, Status, Context, Decision, Consequences, and References.
  - Date and author or originating role (Orchestrator, Auditor, etc.).
- ADRs **MUST** reference corresponding workflow or audit logs when applicable.
- Superseded ADRs **MUST** be marked `Deprecated` but retained for historical integrity.
- ADRs **SHOULD** be linked from the Method Spec or README where directly relevant.

4.5 Manifest and Run Directory Requirements

Each repository **MUST** include a `/runs/` directory containing subfolders for every attested execution.
Each Run folder **MUST** contain:

File	Description	Requirement
`manifest.json` or `.md`	Falsifiability declaration and preconditions for the run.	MUST
`report.md`	Primary human-readable research output.	MUST
`approval.json`	Attestation record confirming verification or rejection.	MUST
`hash.txt` or inclusion in `SHA256SUMS.txt`	Integrity signature of run artifacts.	MUST
`metadata.json`	Contextual parameters, participants, and timestamps.	SHOULD

All files within a Run folder **MUST** be immutable once signed and referenced in `SHA256SUMS.txt`.

4.6 Integrity and Attestation

- The repository **MUST** maintain a single authoritative checksum file (`SHA256SUMS.txt`) in the root directory.

- Every attested artifact (PDF, manifest, run report, ADR, etc.) **MUST** be listed with its SHA-256 digest.
- Attestation signatures **MUST** follow the cryptographic signing policy defined in ADR-0015.
- Human signoffs **MUST** reference ADR-0012 and be recorded in `/logs/overrides/` if manual validation was required.

4.7 Documentation & PDF Builds

- All core documents (`AWO_Method_Spec`, `AWO_Whitepaper`, `AWO_Adoption_Guide`) **MUST** be compiled via automated workflows.
- The build system **MUST** ensure reproducibility and checksum verification per ADR-0016.
- Generated PDFs **MUST** reside in `/docs/` and be referenced in the release assets.

4.8 Governance and Continuity

- The repository **MUST** include a governance note or `README` section referencing ADR-0017, confirming oversight under the Aurora Research Initiative (ARI).
- Any repository transfer, rename, or fork **MUST** preserve ADR continuity and integrity hashes.
- The repository's `README.md` **MUST** declare the canonical DOI (see ADR-0010).

4.9 Compliance Tiers (Informative)

AWO compliance operates in three tiers, as detailed in the Adoption Guide: - **Minimum Compliance** — manual logging and attestation only.
- **Standard Compliance** — includes structured manifests, checksums, and ADR linking.
- **Full Compliance** — includes automated builds, schema validation, and cryptographic attestation.

4.10 Future Integration

When CRI-CORE enforcement becomes active: - Validation schemas in `/schemas/` **WILL** become executable policies. - Manual override logs in `/logs/overrides/` **WILL** trigger runtime verification events. - Repository audits **WILL** be automatically generated from `SHA256SUMS.txt` diffs.

TODO: Link this section to CRI enforcement spec once published.

---

5. Lifecycle and Run Phases

5.1 Overview

Every AWO-compliant project advances through a reproducible four-phase lifecycle.
These phases define the canonical order of epistemic operations, ensuring that each claim moves from **hypothesis** to **verified artifact** under transparent governance.

The canonical lifecycle phases are:

1. **Fan-out (Planning)** — Definition of hypotheses, falsifiability conditions, and manifests.

2. **Consensus (Execution)** — Generation of reasoning paths or experimental runs.

3. **Attestation (Verification)** — Evaluation of results against falsifiability and audit criteria.

4. **Archival (Publication)** — Finalization, signing, and release of immutable artifacts.

Each phase yields its own artifacts, logs, and ADRs, forming a complete reasoning lineage.

5.2 Phase 1 — Fan-Out (Planning)

Purpose To define *what will be tested, how it could fail,* and *who will oversee verification.*
Fan-out begins the epistemic process by expanding a single research goal into a set of structured, falsifiable hypotheses.

Activities

- Create or update the **Run Manifest** (`manifest.md` or `.json`) describing:

  - Objective, assumptions, and falsifiability criteria.
  - Expected inputs, data sources, and transformation paths.
  - Defined roles (Orchestrator, Evaluator, Auditor, Synthesizer).
- Register a new **ADR** if the planned run changes methodology or assumptions.
- Log planning steps under `/logs/workflow/`.

Inputs

- Prior ADRs and manifests.
- Source data, context from previous runs, or external citations.

Outputs

- Updated or new manifest.
- Associated ADR (e.g., "ADR-NNNN — Run Plan vX").
- Planning log entries.

Trigger for Next Phase Orchestrator approval of the manifest and human sign-off per **ADR-0012 (Human-in-Loop Validation)**.

5.3 Phase 2 — Consensus (Execution)

Purpose To perform reasoning, model inference, or experimental execution under the conditions defined in the manifest.

Activities

- Execute all reasoning agents or models specified.
- Collect generated outputs, intermediate data, and system logs.
- Optionally employ multiple agents or parameter sweeps to create a **fan-out of reasoning paths.**
- Evaluate preliminary consistency via internal scoring or evaluator votes.
- Record all contextual metadata (versions, seeds, hashes) in `/runs/<RUN_ID>/metadata.json`.

Inputs

- Manifest and ADR definitions.

- Roles configuration file (implicit or explicit).
- Versioned environment and model parameters.

Outputs

- Raw results and intermediate artifacts.
- Execution logs (`/logs/workflow/`).
- Temporary evaluation summaries.

Trigger for Next Phase Evaluator consensus or Orchestrator decision to proceed to formal verification.

5.4 Phase 3 — Attestation (Verification)

Purpose To formally verify that results meet falsifiability and audit criteria defined in the manifest.
This phase converts raw outputs into **attested knowledge.**

Activities

- Auditors perform validation checks:
  - Schema compliance (structure, completeness).
  - Logical falsifiability (did any counterexample occur?).
  - Provenance linkage (data lineage intact).
- Record results in `/logs/audits/`.
- If human validation is required, document it in `/logs/overrides/` referencing **ADR-0012.**
- Generate `approval.json` containing:
  - Verdict (`approved`, `rejected`, or `needs-revision`).
  - Auditor signatures or cryptographic attestations (per **ADR-0015**).
  - References to manifest, run hash, and ADR numbers.

Inputs

- Run artifacts from Phase 2.
- Manifest and corresponding ADRs.

Outputs

- `approval.json` (attestation record).
- Audit log entries with validation outcomes.
- Updated `SHA256SUMS.txt`.

Trigger for Next Phase All required approvals recorded and checksums generated.

5.5 Phase 4 — Archival (Publication)

Purpose To freeze, sign, and publish verified artifacts for long-term reproducibility and citation.
This is the point at which a Run becomes an immutable element of the research record.

Activities

- Move verified run artifacts into `/runs/` and compute final checksums.
- Update `SHA256SUMS.txt` and verify integrity.
- Generate or update PDFs via automated workflows (per **ADR-0016**).
- Create changelog entry summarizing the Run and resulting ADR references.
- Tag repository version (e.g., `v1.2.1`) and attach signed artifacts.
- Register DOI once repositories are synced to Zenodo or equivalent archival service (per **ADR-0010**).

Inputs

- Verified artifacts from Attestation.
- Final audit results and approval files.

Outputs

- Immutable run directory with all signatures.
- Release tag and checksum record.
- DOI-linked archival snapshot.

Trigger for Completion Publication of DOI and confirmation of checksum match against `SHA256SUMS.txt`.

5.6 Lifecycle Transition Matrix

Phase	Primary Inputs	Primary Outputs	Responsible Roles	Key Artifacts	Governing ADRs
Fan-Out	Prior ADRs, data, goals	Manifest, planning log	Orchestrator, Critic	`manifest.json`, ADR-NNNN	0002, 0009, 0012
Consensus	Manifest	Raw results, metadata	Evaluator, Synthesizer	`metadata.json`, temp logs	0002, 0013
Attestation	Run outputs	`approval.json`, audit logs	Auditor, Orchestrator	`/logs/audits/`, `/logs/overrides/`	0003, 0012, 0015
Archival	Verified artifacts	Signed release, DOI, checksums	Orchestrator, Auditor	`SHA256SUMS.txt`, release tag	0010, 0014, 0016, 0017

5.7 Compliance Expectations

- Every AWO Run **MUST** pass through all four phases in order.
- No phase **MAY** be skipped or merged unless justified in an ADR and recorded in `/logs/overrides/`.
- Each transition **MUST** be timestamped and logged.
- Failing a phase (e.g., rejection in Attestation) **MUST** result in either iteration or termination — never silent acceptance.
- Archival freezes all prior phases; no edits are permitted post-checksum.

5.8 Future Automation Notes

Once CRI-CORE is operational: - Each phase will map to a discrete enforcement module.
- State transitions will be validated via CRI schema events.
- Overrides will trigger automated diff-based verification alerts.

TODO: Define JSON schema alignment between lifecycle phases and CRI runtime once available.

---

6. Artifacts and Provenance Rules

6.1 Purpose

This section defines the mandatory artifacts, metadata files, and validation mechanisms that constitute the **evidence chain** in every AWO-compliant repository.
All artifacts must be uniquely identifiable, cryptographically verifiable, and cross-linked to the corresponding log and ADR entries.

6.2 Required Artifacts per Run

Every Run **MUST** produce a verifiable and complete set of artifacts:

File	Description	Required	Notes
workflow_frozen.json	Snapshot of executed parameters, configuration state, and environment context.	Yes	Must be generated immediately before execution.
report.md	Narrative or analytical summary describing the run outcome, metrics, and interpretations.	Yes	May be human- or model-authored but must include run ID and timestamp.
approval.json	Signed attestation record confirming human or automated validation per ADR-0012.	Yes	Must reference corresponding manifest and checksum hashes.

File	Description	Required	Notes
SHA256SUMS.txt	Integrity registry listing all artifact hashes within `/runs/` and `/docs/`.	Yes	Updated after each attested run.
manifest.json or **manifest.md**	Defines falsifiability boundaries, inputs, and expected failure conditions.	Yes	Must be versioned and cross-referenced in ADRs and logs.

All files above **MUST** exist in each run folder (`/runs/<RUN_ID>/`).
All entries **MUST** be immutable once signed and referenced in `SHA256SUMS.txt`.

6.3 File Naming and Structure Conventions

- Each run directory **MUST** be timestamped or uniquely identified (e.g., `RUN_2025-10-28_001`).
- Filenames **MUST** use lowercase alphanumeric characters and underscores only.
- Each file **MUST** contain a metadata header (JSON or YAML front-matter) including:
  - Run ID
  - Timestamp (ISO 8601)
  - Origin role (Orchestrator, Auditor, etc.)
  - Linked ADR IDs
  - Provenance lineage references (see below)

6.4 Provenance Chain and Lineage Requirements

The **Provenance Chain** represents the traceable path connecting: 1. Manifest → Workflow execution → Report → Approval → Archive.
2. ADR decisions and logs that define or verify each step.

Minimum Provenance Links

Relationship	Requirement
Manifest Report	The report **MUST** reference the manifest version and falsifiability clause.
Report Approval	The approval record **MUST** include hash references of the report and manifest.
Approval SHA256SUMS	Each approval **MUST** verify against current checksum state.
SHA256SUMS Release	The release process **MUST** include and verify the checksum file.
ADR All	Relevant ADR numbers **MUST** be cited in manifest, report, and approval metadata.

All provenance references **MUST** be machine-readable and auditable via JSON key paths or Markdown tables.

6.5 Validation and Versioning Rules

- Artifacts **MUST** conform to JSON or Markdown schemas defined under `/schemas/`.
- Schema validation **SHOULD** be performed manually for Minimum Compliance, and automatically for Full Compliance (see Adoption Guide).
- Each artifact revision **MUST** be versioned using semantic tags (`vX.Y.Z`).
- Any change that affects results **MUST** trigger a new Run folder.
- Historical artifacts **MUST NOT** be altered; corrections require a superseding Run ID and cross-reference.

6.6 Cryptographic and Attestation Linkage

- Every artifact **MUST** be signed or hashed according to ADR-0015.
- The signing authority (human or model) **MUST** be recorded in `approval.json` and linked to `/logs/overrides/` if any manual intervention occurred.

- Attestation hashes **MUST** match those in `SHA256SUMS.txt`; discrepancies trigger audit flags.

6.7 Integration with Evidence Registry (ADR-0002)

- Each artifact **MUST** register its existence in the Evidence Registry table (maintained under `/docs/Evidence_Registry.md` or equivalent).
- The registry **MUST** include:
  - Artifact path
  - Type (manifest, report, etc.)
  - Linked ADRs
  - Hash value
  - Attestation reference
- Each registry update **MUST** be recorded in `/logs/workflow/` with a unique entry ID.

6.8 Future CRI-CORE Hooks

Once CRI-CORE enforcement is active: - Each artifact type will correspond to a schema module (e.g., `manifest.schema.json`, `approval.schema.json`).
- Provenance chains will be validated automatically using CRI runtime modules.
- Manual overrides will trigger provenance-diff checks to confirm trace continuity.

TODO: Define schema references and CRI module mappings once CRI-CORE Specification v0.1 is published.

---

7. Compliance Language

7.1 Purpose

This section defines the normative language used throughout the Aurora Workflow Orchestration (AWO) specification.
The following terms establish the required, recommended, and optional behaviors that determine conformance.

7.2 Normative Terms

Keyword	Meaning	Enforcement Implication
MUST	A requirement that is absolutely mandatory for AWO compliance. Implementations lacking this behavior are non-conformant.	Hard validation — failure blocks attestation or release.
SHOULD	A strong recommendation. Equivalent alternatives are permitted **only** if explicitly documented and justified in logs or ADRs.	Soft validation — warning status logged; manual review required.
MAY	An optional or discretionary behavior that does not affect compliance.	Informational — no enforcement.

7.3 Interpretive Rules

- The words **MUST**, **MUST NOT**, **SHOULD**, **SHOULD NOT**, and **MAY** are to be interpreted as described in **RFC 2119 / ISO IEC TR 29110**.

- All **MUST** clauses are binding for compliance certification under the Aurora Research Initiative (ARI).

- A **SHOULD** clause may be overridden **only** through a logged exception referencing its justification (see `/logs/overrides/`).

- A **MAY** clause introduces permitted flexibility and cannot be used to claim non-compliance of another implementation.

- Deviations from any **MUST** clause **MUST** be documented as a non-conformance record.

7.4 Mapping of Compliance Levels

Section	Description	Compliance Level
§3 Roles and Responsibilities	Role declaration and separation of duties	**MUST**
§4 Repository Requirements	Directory structure, ADR layout, log subfolders	**MUST**
§5 Lifecycle and Run Phases	Sequential execution order and audit flow	**MUST**
§6 Artifacts and Provenance Rules	Artifact creation, immutability, hash verification	**MUST**
§8 Accountability Matrix	Role–artifact responsibility mapping	**SHOULD**
§9 Versioning and Reproducibility	Tagging, checksum maintenance, archival	**MUST**
§10 Licensing and Attribution	License files and acknowledgments	**MUST**
§11 Adoption and Compliance Tiers	Implementation depth and flexibility	**MAY/SHOULD**
§12 Future Integration	CRI-CORE hooks and schema mappings	**MAY**

7.5 Non-Conformance Handling

- Any violation of a **MUST** clause **MUST** be treated as a **non-compliance event** and logged in `/logs/audits/`.

- Deviations from a **SHOULD** clause **MUST** be recorded in `/logs/overrides/` with justification.

- Repeated or uncorrected non-conformances **MUST** trigger re-attestation or run invalidation.

- Compliance auditors **MAY** issue a variance report summarizing exceptions and resolutions.

7.6 Certification and Conformance Evidence

- A repository claiming AWO compliance **MUST** include a `COMPLIANCE.md` file or equivalent table summarizing clause-level adherence.

- The file **SHOULD** include references to specific ADRs, manifests, and run IDs verifying each claim.

- Attestation signatures in `approval.json` serve as binding statements of compliance at the time of release.

---

8. Roles–Artifact Accountability Matrix

8.1 Purpose

This section defines the accountability relationships between AWO roles (as specified in §3) and the artifacts produced during the lifecycle (as specified in §6).
The objective is to ensure that every file, log, and decision is traceable to a responsible role and attested according to the AWO governance standard.

Each artifact **MUST** have: - A clearly declared **origin role** (who created it).
- A **reviewing or attesting role** (who verified it).
- A **governing ADR reference** defining the applicable rules.

8.2 Role–Artifact Responsibility Matrix

Artifact	Origin Role(s)	Reviewing Role(s)	Governing ADRs	Compliance Level
manifest.json / manifest.md	Orchestrator	Auditor	0002, 0012	**MUST**
workflow_frozen.json	Orchestrator, Evaluator	Auditor	0002, 0004	**MUST**
report.md	Synthesizer	Critic (optional), Auditor	0009, 0012	**MUST**
approval.json	Auditor	Orchestrator (acknowledgment)	0012, 0015	**MUST**
SHA256SUMS.txt	Orchestrator	Auditor	0015, 0016	**MUST**

Artifact	Origin Role(s)	Reviewing Role(s)	Governing ADRs	Compliance Level
ADR files	Orchestrator, Auditor	Orchestrator	0001–0017	**MUST**
/logs/workflow/	Orchestrator	Auditor	0004	**MUST**
/logs/audits/	Auditor	Orchestrator (review only)	0003, 0013	**MUST**
/logs/overrides/	Orchestrator (manual intervention)	Auditor	0004, 0012	**SHOULD**
/schemas/	Orchestrator, Auditor	CRI validator (future)	0002, 0015	**MAY**
/templates/	Orchestrator	N/A	0011	**MAY**
/figures/	Orchestrator	N/A	0009	**MAY**

8.3 Chain of Custody

All artifacts **MUST** maintain a documented chain of custody that records: - **Creation timestamp**
- **Responsible role**
- **Verification signature or attestation hash**
- **Referenced ADR(s)**
- **Linked run ID**

This metadata **MUST** be stored in either: - File front matter (for Markdown-based artifacts), or
- Embedded JSON keys (for structured data).

Example metadata block:

```
run_id: RUN_2025-10-28_001
origin_role: Orchestrator
verified_by: Auditor
adr_refs: [0002, 0012]
sha256: "2f7b3e8e..."
timestamp: 2025-10-28T18:21:00Z
```

8.4 Attestation Logic

1) Primary Attestation

-Each artifact requiring human or automated approval (e.g., approval.json, report.md MUST be signed off by the Auditor role.
-Signatures may be human-readable (signed-by) or cryptographic (per ADR-0015).

2) Secondary Acknowledgment

-The Orchestrator SHOULD acknowledge attested artifacts via changelog or run note entry.
-This creates a closed validation loop and allows two-party accountability.

3)Override Case

-If the Orchestrator bypasses or modifies an attested artifact, an entry MUST be logged in /logs/overrides/ citing justification and relevant ADR(s).

8.5 Accountability Validation (Automated and Manual)

-Automated systems SHOULD validate that every artifact in /runs/ and /docs/ has both an origin and attesting role recorded.
-Manual audits MUST confirm that metadata matches recorded logs and ADRs.
-Missing or ambiguous role assignments MUST trigger a non-conformance flag under §7.5.

8.6 Role Coverage Summary

Role | Primary Responsibilities | Secondary Responsibilities | Key Compliance Points | Orchestrator Manages run lifecycle, produces manifests and workflow logs, coordinates attestation. | Reviews audits, ensures completeness. §3.2, §4.2, §5, §6 | Evaluator | Generates and compares outputs from reasoning models. | Assists in workflow_frozen capture. | §3.2, §5.3 | Auditor | Performs formal verification, approves or rejects artifacts, maintains audit logs. | Validates checksum and signature integrity. | §3.2, §5.4, §6.6 | Synthesizer | Produces consolidated reports from approved reasoning paths. | Supports narrative generation for publication. | §3.2, §5.3 | Critic / Red Team | Optionally challenges claims to test falsifiability. | N/A | §3.2, §5.2 |

8.7 Conformance Evidence

To demonstrate role–artifact compliance:
-Repositories MUST maintain a ROLE_ATTESTATION.md or equivalent manifest summarizing each role's contributions.
-The file SHOULD be updated per release tag and reference ADRs, Run IDs, and hash values.
-Future CRI-CORE integrations MAY automate this process using agent-based signature validation.

8.8 Future Integration Notes

Once CRI-CORE is active:
-Each role's attestation will correspond to a schema validator module (e.g., auditor.schema.json).
-The Accountability Matrix will be machine-enforced through the CRI runtime

layer.
-Non-human agents (models) will sign their outputs using embedded identity tokens or deterministic cryptographic fingerprints.

TODO: Define CRI-CORE accountability schema references upon release of CRI Specification v0.1.

---

8. Governance and Attestation

Each run requires human or automated attestation of validity and completeness.

Core Requirements: - Runs MUST include `approval.json` with reviewer signature and timestamp.
- Attestation MAY include checksum verification and peer confirmation.
- Failed attestations MUST be logged under `/logs/attestation_failures/`.

TODO: Specify acceptable digital signature methods and verification workflows.

---

9. Release and Versioning

AWO-compliant repositories MUST version all outputs and preserve immutability.

Release Requirements: - Each release corresponds to a reproducible state of the repository.
- Tags MUST follow semantic versioning (e.g., `v1.2.1`).
- Releases MUST attach PDF artifacts, SHA256SUMS, and ADR references.
- Released runs MUST NOT be altered post-publication.

TODO: Add instructions for checksum regeneration and Zenodo linkage.

---

10. Licensing and Attribution

AWO uses dual licensing to separate executable and textual components.

- **Code:** Licensed under Apache 2.0.
- **Documentation:** Licensed under CC BY 4.0.
- Attribution MUST include author, ORCID, and concept DOI in derivative works.

TODO: Add structured attribution metadata schema reference.

---

11. Falsifiability Manifests

Each experiment MUST include a falsifiability manifest before execution.

Manifest Contents: - Hypothesis statement
- Predicted outcomes
- Disproof criteria
- Experimental plan
- Acceptance thresholds
- Known risks

TODO: Formalize manifest schema for CRI-CORE parsing.

---

12. Conformance Checklist

Each repository MUST pass the following before claiming AWO compliance:

- ☐ Standard directory structure present.
- ☐ At least one signed run in `/runs/`.
- ☐ ADRs and falsifiability manifests linked.
- ☐ SHA256SUMS.txt present at root.
- ☐ PDF artifacts built successfully.
- ☐ CHANGELOG includes version reference.
- ☐ README links to Whitepaper, Method Spec, Adoption Guide.

TODO: Add automated compliance script references (future CRI module).

---

13. Appendix C — Rationale Summary (Reserved)

TODO: When the Method Spec text is finalized, reintroduce Appendix C summarizing why each rule exists in concise bullet form.
(Placeholder retained for structural continuity.)

End of Specification — Aurora Workflow Orchestration (AWO) v1.2.1 Scaffold