RHUMB WORKFLOW PROTOCOL™ PAPER

Rhumb Workflow Protocol™ Specification Reading Bundle

A curated path through the canonical protocol repository

RWP™ 0.31.0 Implementers, architects, standards reviewers

Contents

PURPOSE

This is a reading packet, not the full spec pasted into a PDF

The protocol repository is the maintained source. This bundle gives implementers an ordered path through the materials so they can understand the contract before building or claiming support.

The practical risk is reading the website first and assuming it is the spec. The website should explain and distribute. The repository specs, schemas, templates, examples, and validator behavior remain authoritative.

READING ORDER

Read the materials in dependency order

Start with how the workflow feels, then move to storage profiles, then the formal contract, then conformance. That order prevents a common implementation mistake: copying a directory tree before understanding what artifact semantics must be preserved.

- `docs/GETTING-STARTED.md`: practical adoption path and first workflow.
- `docs/IMPLEMENTATION-PROFILES.md`: core file tree, Meridian reference profile, and custom profile rules.
- `docs/PROTOCOL.md`: full protocol specification.
- `spec/conformance-levels.md`: required, recommended, and optional support levels.
- `docs/CONFORMANCE.md`: validator behavior, mark-use gate, and current Meridian reference-run status.
- `spec/lifecycle/idea-lifecycle.spec.md`: IDEA state machine.
- `spec/sequence.grammar` and `spec/sequence-parser.md`: phase sequence syntax.
- `templates/` and `examples/`: implementation fixtures.
- `integrations/`: tool-specific adapters after the core is understood.

NORMATIVE VERSUS ADVISORY

Separate protocol requirements from implementation advice

Schemas, grammar, lifecycle specs, and conformance behavior are the closest thing to normative material in the current repository. Guides, diagrams, papers, website pages, and templates explain or instantiate the contract.

Templates are important because implementations actually copy them, and Meridian™ can load them. But templates are still not a license to smuggle Meridian-only runtime behavior into core Rhumb.

ARCHITECTURE PATH

IDEA -> AVD -> ACS -> MP is the protocol method path

The architecture path lets teams start with lightweight idea capture, promote coherent work into an architecture vision, define component contracts, and then execute a managed plan. It is the bridge between casual AI prompting and durable architecture governance.

For YAKKL® Meridian™, that path matters because persona-based interfaces can hide or reveal complexity while still emitting the same protocol artifacts.

COMPATIBILITY

Implementation alignment is a profile and reconciliation problem

Any reference implementation will expose drift between the living protocol and a shipped product. The correct resolution is to synchronize core protocol artifacts, publish explicit profiles, and update implementations when protocol changes harden. For example, Meridian™ (the current reference implementation) has gone through this reconciliation process as RWP™ has evolved.

External implementers should not need any reference implementation's internals to use Rhumb™. They should need only the core contract and, optionally, a profile if they want interoperability with a specific implementation.

VALIDATION CHECKLIST

A first implementation is not done until these checks pass

A tool that merely creates a `PLAN.md` is useful, but it should not claim the same support level as a validator-backed implementation. Declare support depth honestly.

- Artifacts parse as YAML/Markdown with expected frontmatter and schema fields.
- Phase identifiers match the sequence grammar and schema pattern.
- Plan and state agree on plan id and current phase where both are present.
- Handoff paths referenced by state or manifest exist.
- Implementation-specific fields are namespaced or documented.
- Profile claims identify whether the tool supports Core File-Tree, Meridian™ Reference, or a custom profile.