7.6.1.2 Guideline Document: The Evolution Principle

Version: 1.0.0 **Date:** 2025-08-11 **Status:** DRAFT

> Purpose: Make **Evolution (EVOL)** the primary internal and external identifier, organizing principle, and strategic driver. Ensure every stakeholder—engineering, operations, finance, partners, and customers—can see, audit, and plan around product generations.

---

1) Scope & Intent

This principle applies to all systems, subsystems, artifacts, and communications across the Sphere project. It operationalizes **Evolution-first**: product generations (**EVOL-00, EVOL-01, …**) are self-contained, auditable capsules that structure work, govern change, and frame expectations.

Outcomes sought:

- **Visibility:** Evolution state is instantly discoverable in code, docs, UI, packaging, and public comms.
- **Order:** Generations partition architectural eras; within a generation, SemVer governs compatible change.
- **Drive:** Generational goals and exit criteria create focus, motivate delivery, and anchor roadmaps.

---

2) What "Evolution" Means

EVOL-XX = Product Generation.

- **Boundary:** A generation encapsulates architecture, interfaces, verification, and operations for its era.
- **Break rule:** A **system-wide architectural break** opens a **new EVOL**. Within an EVOL, incompatible but scoped changes may increment **MAJOR** (SemVer) without starting a new generation.
- **Artifacts per EVOL:** Charter, architecture/ADR index, specs & ICDs, tests/V&V, ops & SOPs, change log, release notes, migration guides, marketing copy, and a signed manifest.

SemVer inside an EVOL:

- **MAJOR:** Incompatible change **scoped to the EVOL** (e.g., an ICD break that does not require a new architecture era).
- **MINOR:** Backward-compatible additions.
- **PATCH:** Editorial/non-semantic fixes.

---

3) Why Evolution-first (Vision)

1. **Internal compass.** Generations focus teams on a clear goal line ("What ships in EVOL-01?"), simplify trade-offs, and enable parallel work on EVOL-N and EVOL-(N+1).
2. **External signal.** Generations are a **customer-facing identity** (like automotive model generations) that set expectations about capability, compatibility, and support windows.

3. **Audit & trust.** Each EVOL is an auditable capsule—design, tests, operations—supporting certifications, safety reviews, and partner due diligence.
4. **Strategic cadence.** Generational milestones drive funding gates, supplier readiness, and ecosystem planning.

---

4) Core Rules (Non-Negotiable)

1. **Badge the generation everywhere.** Use EVOL labels in filenames, repo paths, binaries, UI About screens, dashboards, API headers, contracts, and marketing.
2. **One EVOL, one SSOT per topic.** Each topic has exactly one APPROVED reference document per EVOL.
3. **New EVOL on architectural break.** If compatibility cannot be maintained across the system boundary (architecture, safety, ops doctrine), you must open **EVOL-(N+1)**.
4. **Freeze, then fork forward.** Freeze EVOL-N (read-only, patch-only) and develop EVOL-(N+1) in a separate capsule. No silent backports across EVOLs.
5. **Traceability is mandatory.** Every artifact in an EVOL links to its RFC/ADR/CR and V&V evidence.

---

5) Lifecycle & Visibility

Lifecycle: Initiate → Work → Release → Freeze & Archive.

- **Initiate:** Write the *EVOL Charter* (scope, goals, compatibility promises, risks, exit criteria). Appoint owners and reviewers.
- **Work:** Produce and evolve all artifacts under `…/7.6.2-evolutions/EVOL-XX/` with CI linting and manifesting.
- **Release:** Tag `EVOL-XX-YYYY.MM`, publish release notes and migration guides, update customer-facing materials.
- **Freeze & Archive:** Move to `…/7.6.3-history/` (read-only). Security and legal notices may update; functionality does not.

Visibility mechanisms (required):

- **current-evolution.md** pointer in the evolutions root.
- **EVOL banner** in user-facing UIs and operational dashboards.
- **Compare pages**: automated diffs EVOL-(N-1) ↔ EVOL-N for key specs and ICDs.
- **Roadmap strip**: Now (EVOL-N), Next (EVOL-N+1), Later (N+2) on the program home page.

---

6) External Identity (Customer-Facing)

Generation labeling:

- Public names include the generation, e.g., *Sphere Earth ONE — EVOL-01*.
- Marketing and documentation lead with the EVOL identity; model-year-style messaging communicates evolution (capabilities, safety level, performance).

Promises per EVOL:

- **Compatibility window:** the minimum duration interfaces will be supported.
- **Support policy:** LTS/maintenance timelines per EVOL.

- **Migration path:** customer-ready guides and tooling from EVOL-(N-1) to EVOL-N.

Automotive analogy (informative): Like BMW model generations, each EVOL is a visible chapter with distinct architecture and capabilities, while trims/options map to MINOR/PATCH evolution within the generation.

---

7) Governance & Decision Criteria

When to open a new EVOL:

- Cross-cutting architectural changes (safety doctrine, structural grid, power topology, thermal envelope, life-support primitives).
- Interface breaks that cannot be shimmied without unacceptable cost or risk.
- Operational model change (e.g., new docking paradigm) that invalidates prior procedures.

Gatekeeping:

- Changes proposing a new EVOL require an **RFC** with impact analysis, migration plan, and customer-facing narrative. A cross-discipline board reviews (Architecture, Safety, Ops, Finance, Programs).

Within-EVOL change:

- Use SemVer and ADR/RFC discipline; default to compatibility, prefer additive designs, and provide deprecation schedules.

---

8) Artifacts & Templates (per EVOL)

Required:

- **EVOL Charter** (scope, goals, risks, exit criteria).
- **Architecture overview** + **ADR index**.
- **SPEC/ICD set** with traceability to requirements and tests.
- **V&V plan & reports**; acceptance evidence.
- **Ops handbook & SOPs**; safety dossier.
- **Change log & release notes**; **migration guide**.
- **Signed manifest** of key artifacts with checksums; EVOL tag.

Template snippets (short):

EVOL Charter (outline)

1. Scope & goals (what this generation must deliver)
2. Compatibility promises (what remains stable; for how long)
3. Risks & mitigations (top 5)
4. Exit criteria for freeze (objective tests & evidence)
5. Timeline: milestones to Release & Freeze

Migration Guide (outline)

1. Audience & prerequisites
2. What changed and why
3. Compatibility matrix (old ↔ new)
4. Step-by-step migration
5. Validation checklist & rollback

---

9) Foldering, Naming & CI Hooks (Summary)

- **Foldering:** Each EVOL lives under `7.6.2-evolutions/EVOL-XX/…`; frozen generations move to `7.6.3-history/EVOL-XX/…`.
- **Naming:** File names carry `<DOC>-<EVOL>-…-v<MAJOR.MINOR.PATCH>-<STATE>.md`. EVOL in path **must** match filename.
- **CI Hooks:** Lint filename ↔ front-matter coherence; generate EVOL tags and manifests; auto-publish compare pages and release notes; block merges on missing RFC/ADR links.

---

10) KPIs & Rituals

KPIs:

- Generation goal completion rate (per milestone).
- Interface stability index (breaks avoided vs proposed).
- Migration lead time for key partners.
- Documentation completeness (SSOT coverage) at Release.

Rituals:

- **EVOL Review** (bi-weekly): status, risks, decision log.
- **Interface Council** (monthly): compatibility & deprecations.
- **Freeze Readiness Review** (gate): verify exit criteria, lock manifests.
- **Customer Briefing** (at Release): public notes, support window, migration aids.

---

11) Non-Goals (to avoid confusion)

- EVOL is **not** a marketing-only label; it reflects real architectural eras.
- EVOL does **not** replace SemVer; it **frames** SemVer within a generation.
- EVOL changes do **not** rewrite history; prior EVOLs remain frozen and auditable.

---

12) Appendix – Quick Reference

Open new EVOL if: architecture or ops doctrine changes system-wide; interfaces cannot be compatibly bridged; safety basis or certification envelope resets.

Stay within EVOL if: change is additive or can be shimmed; risk and cost of migration exceed benefit; safety and ops doctrine remain stable.

Always do: badge the generation, keep one SSOT per topic per EVOL, trace every change, publish migration paths, and freeze the past before building the future.