Origin-Axiom Phase 5:
Program Interface and Viability Dashboard (Rung 0–1 Skeleton)

Origin-Axiom Collaboration

January 8, 2026

Abstract

Phase 5 is a meta-level program phase. It does not introduce new physics assumptions, but instead formalizes an interface over the locked outputs of Phases 3 and 4 and prepares a viability dashboard that can be extended in later rungs. This skeleton document records the scope, non-claims, and interface contracts used by the Phase 5 code, and it provides placeholders for future diagnostics and comparisons once additional data and analysis are brought online.

Contents

1 Preamble **1**

2 Scope and Non-Claims **2**
2.1 Scope 2
2.2 Non-Claims 2

3 Interface Contract **2**
3.1 Configuration 3
3.2 Diagnostics Summary 3

4 Interface-Level Viability Dashboard (Rung 2 Skeleton) **3**

5 Limitations and Next Rungs **4**
5.1 Limitations 4
5.2 Next Rungs (Conceptual) 5

A Interface Claims Table (Stub) **5**

B Reproducibility Notes (Stub) **5**

1 Preamble

Phase 5 is designed as a program-level interface and dashboard phase for the Origin-Axiom project. In contrast to Phases 0–4, which each carry their own technical scopes and claims, Phase 5 is intentionally *derivative*: it consumes already-locked outputs and exposes them in a structured, inspectable way for future synthesis and comparison.

This document is a Rung 0–1 skeleton. It focuses on:

- pinning down the scope and non-claims of Phase 5,
- documenting the interface contract between Phase 5 and the Phase 3/4 outputs that have already been generated, and
- outlining how these ingredients can later be turned into a program-level dashboard and meta-analysis layer.

No new physical hypotheses are introduced here. The goal is to make the existing machinery as transparent, reproducible, and extensible as possible.

2 Scope and Non-Claims

2.1 Scope

The scope of Phase 5, at this skeleton stage, is purely programmatic:

- It reads a configuration file that specifies which Phase 3 and Phase 4 outputs are required and where they live in the repository.
- It runs an interface script that verifies that these assets exist, reports their sizes, and records this information in a machine-readable summary.
- It treats the Phase 3 and Phase 4 outputs as *locked inputs*: Phase 5 may reason about them, but does not modify or regenerate them.

Concretely, the current implementation is centered on:

- the configuration file `phase5/config/phase5_inputs_v1.json`, and
- the summary written by `phase5/src/phase5/phase5_interface_v1.py`, namely `phase5/outputs/tables/phas`

2.2 Non-Claims

At this rung, Phase 5 explicitly *does not*:

- assert any new physical explanation of the cosmological constant, vacuum structure, or field content;
- introduce new parameter fits or change the numerical values produced by Phases 3 and 4;
- draw quantitative cosmological conclusions from external data sources;
- make any statements about observational viability beyond what is already locked into the upstream phases.

All interpretive, explanatory, or phenomenological claims remain anchored in the earlier phases. Phase 5 is an interface and program layer, not a new theoretical rung.

3 Interface Contract

The core of Phase 5 is an explicit, machine-readable contract describing which upstream artifacts it consumes and how they are organized. This contract is encoded in the JSON file `phase5/config/phase5_inputs_v1.json` and enforced by the Python interface script `phase5/src/phase5/phase5`

3.1 Configuration

The configuration file enumerates three logical groups:

- **Phase 3 block:** references to the mechanical baseline diagnostics and measure statistics, such as `phase3/outputs/tables/mech_baseline_scan_diagnostics.json`, `phase3/outputs/tables/phase3_measure` `phase3/outputs/tables/phase3_measure_v1_hist.csv`, and `phase3/outputs/tables/phase3_instability_`
- **Phase 4 block:** references to the F1 mapping diagnostics and FRW toy/probe outputs, including the F1 sanity and shape diagnostics and the various FRW viability, LCDM, shape, and data probes and their masks.
- **External block:** an optional entry for a binned FRW distance dataset, `phase4/data/external/frw_distance_` which may or may not be present in a given copy of the repository.

The configuration does not alter these paths; it records them so the interface can systematically check their presence.

3.2 Diagnostics Summary

Running the interface script produces a summary JSON file:

`phase5/outputs/tables/phase5_interface_v1_summary.json`

which reports, for each configured path:

- the absolute and relative path,
- whether the file exists,
- its size in bytes (if present), and
- any additional notes for non-path entries (such as the external-data description).

In particular, the current summary explicitly confirms that all Phase 3 and Phase 4 inputs required by the configuration are present, and that the external FRW distance file is missing but treated as optional. This is not a physical statement about the data; it is a documentation of the repository state and the robustness requirements imposed on Phase 5.

4 Interface-Level Viability Dashboard (Rung 2 Skeleton)

At this stage, the "viability dashboard" implemented by Phase 5 is deliberately minimal and purely programmatic. Its role is not to define new physical metrics or scores, but to present a compact, machine- and human-readable view of the Phase 3/4 artifacts that the interface sees as inputs.

The script

`phase5/src/phase5/make_interface_dashboard_v1.py`

takes as input the interface summary JSON produced by Rung 0,

`phase5/outputs/tables/phase5_interface_v1_summary.json`,

and emits a flattened CSV

`phase5/outputs/tables/phase5_interface_dashboard_v1_summary.csv`.

Each row of this CSV corresponds to a single entry in the interface configuration (for example, a Phase 3 table or a Phase 4 FRW diagnostic file). The columns are purely structural:

- **section**: a coarse label such as `phase3`, `phase4`, or `external`;
- **key**: the short identifier used in the configuration (e.g. `mech_baseline_diagnostics`, `frw_shape_probe`);
- **relpath**: the repository-relative path, when the entry corresponds to an on-disk artifact;
- **exists**: a boolean indicating whether the file is present on disk;
- **size_bytes**: the file size in bytes (when the file exists);
- **is_path**: a flag distinguishing path-like entries from purely descriptive notes;
- **note**: an optional text annotation for non-path entries (for example, to document that a dataset is external and optional).

In this Rung 2 skeleton, the dashboard is intentionally limited to reporting these structural facts. It does *not*:

- introduce any new derived quantities or scores,
- aggregate diagnostics into a single "viability" number, or
- perform comparisons against external cosmological datasets.

For convenience, the column structure can be summarized schematically as follows:

section	key	relpath	exists	size_bytes	is_path	note
... rows populated by `make_interface_dashboard_v1.py` ...						

Future rungs of Phase 5 may refine this dashboard by:

- introducing explicit, phase-aligned summary metrics derived from existing diagnostics;
- adding small LaTeX tables or figures that present these metrics in a compact way; and
- wiring selected summary artifacts into the unified program-level dashboard discussed in the planning documents.

Any such extensions must remain honest about what is computed where and must not silently re-fit or reinterpret the upstream phases.

5 Limitations and Next Rungs

5.1 Limitations

The current Phase 5 implementation is intentionally narrow:

- It does not yet integrate into the unified paper build pipeline as a canonical Phase 5 PDF artifact.
- It does not present numerical results, plots, or summary statistics beyond the raw interface diagnostics.

- It does not define new claims, hypotheses, or falsifiable predictions on top of Phases 3 and 4.
- It does not ship with external datasets; the FRW distance file, if used, is explicitly optional and may be absent from a clean clone of the repository.

These limitations are consistent with the role of this document as a skeleton: it is preferable to under-claim and then move carefully to stronger rungs.

5.2 Next Rungs (Conceptual)

Possible next rungs for Phase 5 include:

- wiring a Phase 5 paper build into a dedicated gate script and, if appropriate, into the unified build driver;
- adding a small set of well-motivated summary tables that pull together Phase 3 and Phase 4 diagnostics without duplicating their internal derivations;
- defining a minimal, explicit set of "Phase 5 claims" that are purely about program structure and reproducibility (for example, that a particular version of the repository exposes a complete and self-consistent contract between phases);
- extending the interface to support multiple mapping families or vacuum constructions while keeping the contracts explicit and versioned.

Each of these steps should be introduced via its own rung, with clear diffs and explicit documentation of what changed and why.

References

A Interface Claims Table (Stub)

This appendix is reserved for a simple claims table once Phase 5 has well-defined, program-level claims of its own (for example, statements about completeness of the interface, or guarantees about how upstream artifacts are exposed).

At the current skeleton stage, Phase 5 makes no new physical claims. Consequently, this appendix is intentionally left as a placeholder.

B Reproducibility Notes (Stub)

This appendix is intended to record:

- the exact commands used to run the Phase 5 interface,
- the expected structure of the `phase5_interface_v1_summary.json` file, and
- pointers to the gate script that enforces the interface in CI or on collaborator machines.

At present, reproducibility is centered on:

```
scripts/phase5_gate.sh
phase5/src/phase5/phase5_interface_v1.py
phase5/config/phase5_inputs_v1.json
phase5/outputs/tables/phase5_interface_v1_summary.json
```

Future rungs should promote this into a fully structured reproducibility appendix, in line with the other phases.