Atlas Verification Specification for Mixture-of-Experts Transformers

Noumena

Abstract

This document is the verification-spec companion to *Atlas Foundations for Mixture-of-Experts Transformers*. The foundations paper defines the ontology: routed computation takes place on an RMS-induced operational geometry, pretraining builds an atlas on that geometry, and ordinary post-training should preserve atlas semantics by default. This document turns that ontology into an auditable contract.

The contract has six layers. First, it defines a single-checkpoint forward-pass and extraction protocol for the observables that instantiate atlas semantics. Second, it defines checkpoint-comparison alignment rules so that cross-checkpoint measurements are unambiguous even when gains, routers, or expert surfaces drift. Third, it defines canonical and diagnostic drift functionals together with the sufficient statistics required for independent re-bootstrap. Fourth, it defines executable failure predicates and intervention requirements. Fifth, it specifies artifact schemas and manifests. Sixth, it absorbs older routing-geometry tests as legacy precursor artifacts rather than treating them as the canonical schema.

This is a verification contract, not a results paper. Its role is to make atlas-preservation claims exact enough to reject, reproduce, or operationalize.

Contents

1 Introduction

Mixture-of-Experts (MoE) transformers expose a measurement problem that ordinary parameter-space bookkeeping does not solve. Once routing is sparse, the model's operational semantics are distributed across at least three surfaces: the normalized state presented to the routed block, the router that assigns chart transitions, and the expert fields that realize chart-local updates [Shazeer et al., 2017, Lepikhin et al., 2021, Fedus et al., 2022]. If post-training damages any one of those surfaces, the resulting regression may appear as forgetting, collapse, or instability even when gross parameter movement is modest.

Relationship to the foundations paper. The companion foundations paper proposed an ontology: RMS normalization induces an approximately spherical operational geometry [Zhang and Sennrich, 2019]; pretraining of a large MoE constructs an atlas on that geometry; and ordinary post-training should preserve atlas semantics by default rather than broadly redefining the atlas interior. That paper was intentionally not a verification paper. It defined the object. The purpose of this document is to define the receipt layer required to measure that object consistently.

Problem statement. An ontology without a verification contract is too easy to instantiate inconsistently. The main ambiguities are not philosophical. They are procedural: which hidden state is anchored across checkpoints, which checkpoint's gain is used when a router is cross-evaluated, what the set margin actually is, which experts are evaluated for chart-content drift, and which sufficient statistics must be stored so that a reported confidence interval can be independently re-bootstrapped. If those details are left implicit, two implementations can claim to measure the same drift functional while in fact measuring different objects.

Scope. This document specifies:

1. the forward-pass and extraction protocol for atlas observables on a fixed probe family;
2. the checkpoint-comparison alignment rules used to compare a base checkpoint θ_0 and a comparison checkpoint θ_1;
3. the canonical and diagnostic drift functionals implied by the atlas ontology;
4. executable failure predicates and intervention requirements;
5. artifact schemas and manifest requirements;
6. the absorption of earlier routing-geometry artifacts into the new contract.

Non-goals. This document does not claim that any particular model already satisfies the contract. It does not prescribe a training recipe, an intervention recipe, or an artifact-production pipeline beyond the minimum required to make verification auditable. It does not replace the foundations paper's ontology, and it does not preserve the older routing-geometry schemas unchanged when those schemas are only partial matches to the new measurement object.

Canonical objects. The contract is built around a fixed probe family, a fixed protected layer set, and a fixed checkpoint pair (θ_0, θ_1). At a minimum, the observables include the canonical pre-gain state, the exact post-gain gate coordinate, the router output, the routed expert set, the set margin, and tangent-visible expert update fields. The central design constraint is that every cross-checkpoint quantity must be auditable back to stored per-window sufficient statistics.

Design principle. The specification distinguishes *canonical* measurements from *diagnostic* decompositions. For example, the canonical semantic boundary and transition drifts compare router semantics at a fixed conceptual point while allowing each checkpoint to use its own gain. Router-only and gain-only variants are still mandatory diagnostics, but they are not the primary preservation object. The same principle appears throughout the spec: one canonical measurement object, accompanied by diagnostic decompositions that explain failure mode.

Why this is necessary. Continual-learning and forgetting work has long shown that regression cannot be understood purely as gross optimization progress [Kirkpatrick et al., 2017, Parisi et al., 2019]. In the MoE setting, that issue is amplified because sparse routing creates a semantic interface whose failure modes are local, discrete, and often boundary-sensitive. The verification contract in this document therefore prioritizes exact semantics, window-level bootstrap validity, and artifact sufficiency over convenience.

Outputs of this spec. The document produces six concrete outputs:

1. an observable extraction contract;
2. alignment rules for checkpoint comparison;
3. drift functionals with required sufficient statistics;
4. failure predicates and intervention requirements;
5. artifact schemas;
6. a legacy-mapping layer that explains how prior routing-geometry receipts do and do not map to the new ontology.

Proposition 1.1 (Six-layer completeness of the verification contract)**.** *A checkpoint-comparison claim is canonically valid under this specification only if all six layers are fixed together:*

1. *single-checkpoint extraction semantics;*
2. *cross-checkpoint alignment semantics;*
3. *canonical and diagnostic drift functionals with stored sufficient statistics;*
4. *executable failure predicates and intervention requirements;*
5. *schema-valid artifact bundles with explicit linkage;*
6. *an explicit legacy-mapping status for any supporting artifact that is not itself canonical.*

Omitting any one layer leaves at least one atlas-preservation claim underdetermined.

Remark 1.2 (Why the contract is layered)*.* The six layers are not administrative duplication. They separate three distinct sources of implementation drift: observable extraction, cross-checkpoint comparison semantics, and evidentiary validity. The specification is intentionally layered so that disagreement at any one of those levels becomes auditable rather than silently absorbed into a single reported metric.

Roadmap. The next sections proceed in contract order: first the single-checkpoint forward-pass and extraction protocol, then the checkpoint-comparison alignment rules, then the drift functionals and their sufficient statistics, then the failure predicates and intervention requirements, then the artifact schemas, and finally the legacy-mapping layer that absorbs older routing-geometry receipts without treating them as one-to-one replacements.

2 Forward-Pass and Extraction Protocol

This section defines the single-checkpoint extraction contract. Its purpose is narrow: given a checkpoint θ, a probe family $\mathcal{P}$, and a routed layer set $\mathcal{L}_{\text{protect}}$, it specifies exactly what a compliant implementation must run and exactly what observables it must record before any cross-checkpoint comparison is attempted.

2.1 Execution unit

Definition 2.1 (Probe window and checkpoint-native execution)**.** *Fix a checkpoint θ, a probe family $\mathcal{P}$, and a routed layer ℓ. A* probe window *is a deterministic token sequence*

$$w = (x_1, \ldots, x_T) \tag{1}$$

with tokenizer semantics fixed by $\mathcal{P}$. Checkpoint-native execution of w under θ means one teacher-forced forward pass of θ on w with deterministic inference settings and with all routed-layer observables extracted from the actual internal tensors produced by that pass.

Requirement 2.2 (Deterministic execution)**.** The extraction pass must be deterministic with respect to model weights, tokenizer, attention masking, expert masks, score precision, and tie-breaking. Any source of stochasticity that can alter router scores or selected experts must be disabled or explicitly versioned in the produced artifact.

Requirement 2.3 (Score precision)**.** All dispatch-score computations used for sparse-routing observables must be carried out in fp32. This includes score tensors used to compute R_k, e_k, e_{k+1}, and m_{set}. Lower-precision score computation is non-compliant for canonical extraction.

2.2 Router-facing observables

Definition 2.4 (Checkpoint-native operational coordinates)**.** *Fix checkpoint θ, probe token t, and routed layer ℓ. Let*

$$z_{t,\ell}^{(\theta)} \tag{2}$$

denote the canonical pre-gain operational state presented to the routed block, and let

$$g_{t,\ell}^{(\theta)} := \Gamma_\ell^{(\theta)} z_{t,\ell}^{(\theta)} \tag{3}$$

denote the exact post-gain gate coordinate seen by the router. The pair $(z_{t,\ell}^{(\theta)}, g_{t,\ell}^{(\theta)})$ is the checkpoint-native coordinate record for (t, ℓ, θ).

Definition 2.5 (Admissible routed expert set)**.** *Fix checkpoint θ, probe token t, and routed layer ℓ. Let*

$$\mathcal{A}_{t,\ell}^{(\theta)} \subseteq \{1, \ldots, E_\ell\} \tag{4}$$

denote the admissible routed expert set*: the experts that remain eligible for sparse competition after applying all deterministic routing masks, group constraints, and shared-expert exclusions required by the exact gate semantics. Shared or always-on experts that do not participate in sparse competition are excluded from $\mathcal{A}_{t,\ell}^{(\theta)}$.*

Definition 2.6 (Exact dispatch scores and top-k object)**.** *On the admissible routed expert set $\mathcal{A}_{t,\ell}^{(\theta)}$, let*

$$s_{t,\ell}^{(\theta)}(e) \tag{5}$$

be the exact fp32 dispatch score assigned by checkpoint θ to expert $e \in \mathcal{A}_{t,\ell}^{(\theta)}$ at gate coordinate $g_{t,\ell}^{(\theta)}$. Let

$$R_k^{(\theta)}(g_{t,\ell}^{(\theta)}) \subseteq \mathcal{A}_{t,\ell}^{(\theta)} \tag{6}$$

be the unordered top-k sparse-routing set, with deterministic ordering and tie-breaking supplied by the exact gate semantics. Let $e_k^{(\theta)}(g_{t,\ell}^{(\theta)})$ and $e_{k+1}^{(\theta)}(g_{t,\ell}^{(\theta)})$ denote the k-th and $(k+1)$-th ordered competitors on the admissible set.

Definition 2.7 (Set margin)**.** *Fix the ordered admissible dispatch scores*

$$s_{(1)}^{(\theta)} \geq \cdots \geq s_{(|\mathcal{A}|)}^{(\theta)} \tag{7}$$

induced by $s_{t,\ell}^{(\theta)}(\cdot)$ on $\mathcal{A}_{t,\ell}^{(\theta)}$. The set margin *is*

$$m_{\text{set},t,\ell}^{(\theta)} := s_{(k)}^{(\theta)} - s_{(k+1)}^{(\theta)}. \tag{8}$$

It is computed after deterministic eligibility masks and exclusions, and before any softmax normalization, post-selection renormalization, or expert-weight normalization.

Requirement 2.8 (Boundary-margin semantics)**.** Any artifact or downstream test that stratifies tokens by routing boundary proximity must use Definition 2.7 unless it is explicitly labeled non-canonical. Probability gaps, normalized score gaps, and pre-mask score gaps are not canonical substitutes for m_{set}.

Definition 2.9 (Router output record)**.** *Fix (t, ℓ, θ). Let*

$$w_{t,\ell}^{(\theta)} \tag{9}$$

denote the exact sparse router output on $g_{t,\ell}^{(\theta)}$, including the sparse active set, routed weights, and any ordering information required to reproduce the checkpoint's dispatch semantics. The tuple

$$\left(z_{t,\ell}^{(\theta)}, g_{t,\ell}^{(\theta)}, s_{t,\ell}^{(\theta)}, R_k^{(\theta)}, e_k^{(\theta)}, e_{k+1}^{(\theta)}, m_{\text{set},t,\ell}^{(\theta)}, w_{t,\ell}^{(\theta)}\right) \tag{10}$$

is the minimum canonical router-facing record for (t, ℓ, θ).

2.3 Expert-field observables

Definition 2.10 (Checkpoint-native pre-expert input map)**.** *Fix checkpoint θ, routed layer ℓ, and expert e. Let*

$$\psi_{e,\ell}^{(\theta)} \tag{11}$$

denote the deterministic checkpoint-native map that sends an operational state to the exact tensor consumed by expert e under checkpoint θ. This map includes any implemented subgraph between the canonical routed state and the expert input, such as gain application, reshaping, expert-specific affine maps, or other deterministic pre-expert transforms.

Definition 2.11 (Checkpoint-native expert update)**.** *Fix checkpoint θ, probe token t, routed layer ℓ, and expert e. The checkpoint-native raw expert update is*

$$u_{e,t,\ell}^{(\theta)} := \mathrm{Expert}_{e,\ell}^{(\theta)}\big(\psi_{e,\ell}^{(\theta)}(z_{t,\ell}^{(\theta)})\big), \tag{12}$$

where the right-hand side denotes the raw residual-branch output of expert e before sparse router weighting and before residual addition. The checkpoint-native tangent-visible update is

$$\widetilde{u}_{e,t,\ell}^{(\theta)} := P\big(z_{t,\ell}^{(\theta)}\big)u_{e,t,\ell}^{(\theta)}, \tag{13}$$

with $P(\cdot)$ denoting the first-order visible projector induced by the normalization geometry at the recorded operational state.

Remark 2.12 (Why the single-checkpoint projector is not the comparison projector)*.* Definition 2.11 defines the checkpoint-native visible field extracted from a single forward pass. Cross-checkpoint chart-content comparison is stricter: the comparison protocol in Section 3 re-anchors expert evaluation on the base canonical state and projects both checkpoints with the same base-anchor projector.

Definition 2.13 (Canonical local expert evaluation set)**.** *For checkpoint-native extraction at (t, ℓ, θ), define the local expert evaluation set*

$$\mathcal{E}_{\mathrm{local},t,\ell}^{(\theta)} := R_k^{(\theta)}(g_{t,\ell}^{(\theta)}) \cup \big\{e_{k+1}^{(\theta)}(g_{t,\ell}^{(\theta)})\big\}. \tag{14}$$

This is the minimal local neighborhood needed to characterize the instantiated routed chart and its nearest boundary competitor.

Requirement 2.14 (Evaluation-mode labeling)**.** Any artifact that records expert-field measurements must declare one of the following evaluation modes:

1. **canonical-local**: evaluate only the set in Definition 2.13;
2. **expanded-local**: evaluate the canonical-local set plus a pre-registered adjacent competitor expansion;
3. **exhaustive**: evaluate all routed experts in the layer.

Only **canonical-local** is canonical for atlas-preservation measurements. Expanded-local and exhaustive measurements are admissible only when explicitly labeled non-canonical diagnostics.

2.4 Stored sufficient statistics

Requirement 2.15 (Per-window sufficiency)**.** A compliant extraction artifact must store per-window sufficient statistics for all observables that enter later bootstrap confidence intervals. Token-level values may be retained, but per-window arrays must be sufficient to recompute every reported window-level summary without rerunning the model.

Requirement 2.16 (Layerwise extraction manifest)**.** For each extracted routed layer, the artifact must record:

1. the exact gate semantics profile and score dtype;
2. the routed-expert eligibility rule, including any shared-expert exclusion;

3. the evaluation mode from Requirement 2.14;
4. the projector convention used for tangent-visible updates; and
5. the exact probe-window sampling and bootstrap seed metadata.

Without this manifest, the extracted observables are not auditable.

3 Checkpoint-Comparison Alignment Rules

This section defines how observables from two checkpoints are paired. The single most important rule is that cross-checkpoint comparison is not performed on arbitrary native states from each checkpoint in isolation. It is performed by anchoring evaluation on a base canonical state and then declaring which checkpoint's weights act on which object.

3.1 Base-anchor alignment

Definition 3.1 (Base and comparison checkpoints)**.** *Fix a base checkpoint θ_0, a comparison checkpoint θ_1, a probe family $\mathcal{P}$, and a protected routed layer set $\mathcal{L}_{\text{protect}}$. For each probe token t and layer $\ell \in \mathcal{L}_{\text{protect}}$, let*

$$z_{0,t,\ell} := z_{t,\ell}^{(\theta_0)} \tag{15}$$

denote the canonical base operational state obtained from checkpoint-native execution of θ_0. This is the base anchor state *for all semantic cross-checkpoint measurements at (t, ℓ).*

Definition 3.2 (Aligned gate coordinates)**.** *At a fixed base anchor state $z_{0,t,\ell}$, define the aligned gate coordinates*

$$g_{0\to0,t,\ell} := \Gamma_\ell^{(\theta_0)} z_{0,t,\ell}, \tag{16}$$

$$g_{0\to1,t,\ell} := \Gamma_\ell^{(\theta_1)} z_{0,t,\ell}. \tag{17}$$

The first is the base checkpoint's exact gate coordinate on its own anchor state. The second is the comparison checkpoint's exact gate coordinate on the same conceptual point*, after applying the comparison checkpoint's own gain.*

Requirement 3.3 (Canonical semantic alignment for router-facing comparisons)**.** Canonical boundary and transition comparisons must be evaluated on the aligned coordinates in Definition 3.2. It is non-compliant to evaluate the comparison router only on $\Gamma^{(\theta_0)} z_{0,t,\ell}$ and call the result the canonical semantic comparison. That frozen-base-gain cross-evaluation is a diagnostic, not the canonical measurement.

3.2 Boundary and transition decomposition

Definition 3.4 (Semantic boundary objects)**.** *Fix (t, ℓ) and the aligned coordinates of Definition 3.2. Define*

$$R_{0\to0,t,\ell} := R_k^{(\theta_0)}(g_{0\to0,t,\ell}), \tag{18}$$

$$R^{\text{sem}}_{0\to1,t,\ell} := R_k^{(\theta_1)}(g_{0\to1,t,\ell}), \tag{19}$$

$$w_{0\to0,t,\ell} := \text{Router}_{\theta_0}(g_{0\to0,t,\ell}), \tag{20}$$

$$w^{\text{sem}}_{0\to1,t,\ell} := \text{Router}_{\theta_1}(g_{0\to1,t,\ell}). \tag{21}$$

These are the canonical semantic boundary and transition objects for comparison of θ_0 and θ_1 at the fixed base anchor state.

Definition 3.5 (Router-only diagnostic objects)**.** *Fix the same (t, ℓ) and hold the base gate coordinate fixed. Define*

$$R^{\text{router}}_{0\to1,t,\ell} := R^{(\theta_1)}_k(g_{0\to0,t,\ell}), \tag{22}$$

$$w^{\text{router}}_{0\to1,t,\ell} := \text{Router}_{\theta_1}(g_{0\to0,t,\ell}). \tag{23}$$

These isolate the effect of changing router-side parameters while suppressing gain-induced motion of the gate coordinate. They are diagnostic-only quantities.

Definition 3.6 (Gain-only diagnostic objects)**.** *Fix the comparison checkpoint router and vary only the aligned gate coordinate. Define the gain-only routed objects*

$$R^{\text{gain}}_{0\to1,t,\ell} := \Big(R^{(\theta_1)}_k(g_{0\to0,t,\ell}),\ R^{(\theta_1)}_k(g_{0\to1,t,\ell})\Big), \tag{24}$$

$$w^{\text{gain}}_{0\to1,t,\ell} := \Big(\text{Router}_{\theta_1}(g_{0\to0,t,\ell}),\ \text{Router}_{\theta_1}(g_{0\to1,t,\ell})\Big). \tag{25}$$

The induced disagreement between the two entries measures gain-interface drift under a fixed comparison router.

Requirement 3.7 (Mandatory three-way router decomposition)**.** Any compliant comparison artifact that reports boundary or transition drift must report all three line items:

1. the canonical semantic quantity based on Definition 3.4;
2. the router-only diagnostic based on Definition 3.5;
3. the gain-only diagnostic based on Definition 3.6.

The semantic quantity is canonical. The router-only and gain-only quantities are mandatory diagnostics whose purpose is to resolve the source of a semantic drift signal rather than to replace it.

3.3 Coordinate alignment and reporting

Definition 3.8 (Checkpoint-native comparison coordinates)**.** *For the same probe token t and routed layer ℓ, let*

$$z_{1,t,\ell} := z^{(\theta_1)}_{t,\ell}, \qquad g_{1,t,\ell} := \Gamma^{(\theta_1)}_{\ell} z_{1,t,\ell} \tag{26}$$

be the comparison checkpoint's own native operational coordinates on the same teacher-forced probe token. Then

$$(z_{0,t,\ell}, z_{1,t,\ell}) \qquad \textit{and} \qquad (g_{0\to0,t,\ell}, g_{1,t,\ell}) \tag{27}$$

are the paired pre-gain and post-gain coordinate records for total checkpoint-native drift.

Requirement 3.9 (Dual coordinate reporting)**.** Any compliant comparison artifact must report both:

1. a pre-gain canonical coordinate line item based on $z_{0,t,\ell}$ and $z_{1,t,\ell}$; and
2. a post-gain gate-coordinate line item based on $g_{0\to0,t,\ell}$ and $g_{1,t,\ell}$.

The post-gain quantity is the implementation-correct router-facing coordinate drift. The pre-gain quantity is required so the contribution of gain anisotropy to coordinate drift is recoverable without recomputation.

3.4 Aligned chart-content extraction

Definition 3.10 (Base-anchor expert input and raw update)**.** *Fix base anchor state $z_{0,t,\ell}$, checkpoint $\theta \in \{\theta_0, \theta_1\}$, and expert e. Define the base-anchor expert input*

$$h^{(\theta)}_{e,t,\ell}(z_0) := \psi^{(\theta)}_{e,\ell}(z_{0,t,\ell}), \tag{28}$$

where $\psi^{(\theta)}_{e,\ell}$ is the checkpoint-native pre-expert input map of Definition 2.10. Define the base-anchor raw update

$$u^{(\theta)}_{e,t,\ell}(z_0) := \mathrm{Expert}^{(\theta)}_{e,\ell}\big(h^{(\theta)}_{e,t,\ell}(z_0)\big). \tag{29}$$

Definition 3.11 (Base-anchor tangent-visible expert field)**.** *Fix the base anchor state $z_{0,t,\ell}$ and define the base-anchor projector*

$$P_{0,t,\ell} := P(z_{0,t,\ell}). \tag{30}$$

For checkpoint $\theta \in \{\theta_0, \theta_1\}$ and expert e, define the aligned tangent-visible expert field

$$\widetilde{u}^{(\theta)}_{e,t,\ell}(z_0) := P_{0,t,\ell}\, u^{(\theta)}_{e,t,\ell}(z_0). \tag{31}$$

Both checkpoints are therefore evaluated on the same base conceptual point and projected through the same first-order visible geometry.

Definition 3.12 (Canonical comparison expert set)**.** *At the base anchor state $z_{0,t,\ell}$, define the canonical comparison expert set*

$$\mathcal{E}_{\mathrm{canon},t,\ell} := R^{(\theta_0)}_k(g_{0\to 0,t,\ell}) \cup \big\{e^{(\theta_0)}_{k+1}(g_{0\to 0,t,\ell})\big\}. \tag{32}$$

This is the local neighborhood of the actually instantiated base atlas at (t, ℓ).

Requirement 3.13 (Canonical content locality)**.** Canonical chart-content drift must be computed on the set in Definition 3.12 unless an expanded-local or exhaustive mode is explicitly declared. Evaluating every routed expert on a base anchor state is permitted only as a labeled non-canonical diagnostic, because it probes off-manifold expert behavior that the instantiated atlas need not realize.

3.5 Comparison manifest

Requirement 3.14 (Alignment manifest)**.** Every comparison artifact must declare:

1. the base and comparison checkpoint identifiers;
2. the protected layer set and probe family identifier;
3. the canonical semantic alignment rule of Requirement 3.3;
4. the three-way router decomposition of Requirement 3.7;
5. the dual coordinate reporting rule of Requirement 3.9; and
6. the expert evaluation mode and expert-set definition used for content extraction.

Without this manifest, two artifacts cannot be assumed to implement the same comparison semantics.

4 Drift Functionals and Sufficient Statistics

This section defines the canonical atlas-drift functionals together with the minimum sufficient statistics required for independent re-bootstrap. The governing rule is simple: all canonical comparisons are made on a fixed probe family, bootstrap over windows rather than tokens, and distinguish *concept drift*, *gain-interface drift*, and *router-core drift* rather than collapsing them into a single un-attributed number.

4.1 Comparison objects

Fix a base checkpoint θ_0, a comparison checkpoint θ_1, a probe family $\mathcal{P}$, and a protected routed layer set $\mathcal{L}_{\text{protect}}$. For a probe token t at layer $\ell \in \mathcal{L}_{\text{protect}}$, let

$$z_{t,\ell}^{(\theta_j)} \in \mathcal{M}_\ell \tag{33}$$

denote checkpoint θ_j's own-forward canonical pre-gain operational state, and let

$$g_{t,\ell}^{(\theta_j)} := \Gamma_\ell^{(\theta_j)} z_{t,\ell}^{(\theta_j)} \tag{34}$$

denote checkpoint θ_j's own-forward exact post-gain gate coordinate.

For base-anchored cross-checkpoint comparisons, define the anchored canonical state

$$z_{0,t,\ell} := z_{t,\ell}^{(\theta_0)}, \tag{35}$$

and the checkpoint-native anchored gate coordinates

$$g_{0\to j,t,\ell} := \Gamma_\ell^{(\theta_j)} z_{0,t,\ell}. \tag{36}$$

Write

$$w_{0\to j,t,\ell} := \text{Router}_{\theta_j}(g_{0\to j,t,\ell}) \tag{37}$$

for checkpoint θ_j's router output on the anchored gate coordinate, and let $R_k^{(\theta_j)}(g)$ denote the unordered top-k routed expert set under checkpoint θ_j's exact gate semantics.

Definition 4.1 (Canonical local expert evaluation set)**.** *For a probe token t and protected layer ℓ, let*

$$e_{k+1;0}(t,\ell) \tag{38}$$

denote the $(k+1)$-st competitor under checkpoint θ_0 evaluated on $g_{0\to 0,t,\ell}$. Define the canonical local expert evaluation set

$$\mathcal{E}_{\text{loc}}(t,\ell) := R_k^{(\theta_0)}(g_{0\to 0,t,\ell}) \cup \{e_{k+1;0}(t,\ell)\}. \tag{39}$$

This is the default expert set used for chart-content drift. Any exhaustive or off-manifold expert evaluation must be explicitly labeled non-canonical.

4.2 Coordinate drift

Coordinate drift is split into a canonical pre-gain channel and an implementation-correct post-gain gate channel. The former isolates transport and concept motion in the canonical state geometry; the latter measures the exact drift seen by the router interface.

Definition 4.2 (Coordinate drift channels)**.** *For a protected layer ℓ, define*

$$\Delta^{z}_{\text{coord}}(\ell) := \mathbb{E}_t\Big[1 - \cos\big(z^{(\theta_0)}_{t,\ell}, z^{(\theta_1)}_{t,\ell}\big)\Big], \tag{40}$$

$$\Delta^{g}_{\text{coord}}(\ell) := \mathbb{E}_t\Big[1 - \cos\big(g^{(\theta_0)}_{t,\ell}, g^{(\theta_1)}_{t,\ell}\big)\Big]. \tag{41}$$

We call $\Delta^{z}_{\text{coord}}$ the pre-gain canonical coordinate drift *and $\Delta^{g}_{\text{coord}}$ the* post-gain gate drift*.*

Reporting rule. Both channels are mandatory outputs. $\Delta^{g}_{\text{coord}}$ is the implementation-correct router-facing coordinate drift; $\Delta^{z}_{\text{coord}}$ is the canonical decomposition that makes gain contribution recoverable without recomputation.

4.3 Boundary drift

Boundary drift is defined on a fixed conceptual point given by the base canonical state $z_{0,t,\ell}$. The canonical semantic form uses checkpoint-native gains on that same conceptual point. Two mandatory diagnostics then separate router-core drift from gain-interface drift.

Definition 4.3 (Boundary drift channels)**.** *For a protected layer ℓ, define the anchored top-k sets*

$$R_{0\to j}(t,\ell) := R^{(\theta_j)}_{k}(g_{0\to j,t,\ell}), \tag{42}$$

$$R^{\text{router}}_{0\Rightarrow 1}(t,\ell) := R^{(\theta_1)}_{k}(g_{0\to 0,t,\ell}). \tag{43}$$

Then define:

1. ***Canonical semantic boundary drift***

$$\Delta^{\text{sem}}_{\text{boundary}}(\ell) := \mathbb{E}_t\left[1 - \frac{|R_{0\to 0}(t,\ell) \cap R_{0\to 1}(t,\ell)|}{k}\right]. \tag{44}$$

2. ***Router-only boundary drift***

$$\Delta^{\text{router}}_{\text{boundary}}(\ell) := \mathbb{E}_t\left[1 - \frac{|R_{0\to 0}(t,\ell) \cap R^{\text{router}}_{0\Rightarrow 1}(t,\ell)|}{k}\right]. \tag{45}$$

3. ***Gain-only boundary drift***

$$\Delta^{\text{gain}}_{\text{boundary}}(\ell) := \mathbb{E}_t\left[1 - \frac{|R^{\text{router}}_{0\Rightarrow 1}(t,\ell) \cap R_{0\to 1}(t,\ell)|}{k}\right]. \tag{46}$$

Interpretation. $\Delta^{\text{sem}}_{\text{boundary}}$ is the canonical preservation quantity. $\Delta^{\text{router}}_{\text{boundary}}$ attributes drift to router-core reinterpretation at fixed base gate coordinates. $\Delta^{\text{gain}}_{\text{boundary}}$ attributes drift to checkpoint-native gain mismatch while holding the comparison router fixed.

Proposition 4.4 (Boundary decomposition is non-substitutable)**.** *At a fixed base anchor state, the semantic, router-only, and gain-only boundary channels answer different questions:*

1. $\Delta^{\text{sem}}_{\text{boundary}}$ *asks whether the active chart neighborhood changed under the full checkpoint-to-checkpoint semantic comparison;*
2. $\Delta^{\text{router}}_{\text{boundary}}$ *asks whether the comparison router reinterprets the base gate coordinate;*
3. $\Delta^{\text{gain}}_{\text{boundary}}$ *asks whether checkpoint-native gain deformation changes the neighborhood while holding the comparison router fixed.*

Consequently, none of the diagnostic channels may replace the semantic channel, and neither diagnostic channel may be inferred from the other.

4.4 Transition drift

Transition drift uses the same anchored comparison structure as boundary drift, but compares router distributions rather than only active sets.

Definition 4.5 (Transition drift channels)**.** *For a protected layer ℓ, define the router outputs*

$$w^{\text{router}}_{0\Rightarrow 1,t,\ell} := \text{Router}_{\theta_1}(g_{0\to 0,t,\ell}). \tag{47}$$

Then define:

1. ***Canonical semantic transition drift***

$$\Delta^{\text{sem}}_{\text{transition}}(\ell) := \mathbb{E}_t\Big[D_{\text{JS}}\big(w_{0\to 0,t,\ell}, w_{0\to 1,t,\ell}\big)\Big]. \tag{48}$$

2. ***Router-only transition drift***

$$\Delta^{\text{router}}_{\text{transition}}(\ell) := \mathbb{E}_t\Big[D_{\text{JS}}\big(w_{0\to 0,t,\ell}, w^{\text{router}}_{0\Rightarrow 1,t,\ell}\big)\Big]. \tag{49}$$

3. ***Gain-only transition drift***

$$\Delta^{\text{gain}}_{\text{transition}}(\ell) := \mathbb{E}_t\Big[D_{\text{JS}}\big(w^{\text{router}}_{0\Rightarrow 1,t,\ell}, w_{0\to 1,t,\ell}\big)\Big]. \tag{50}$$

Interpretation. Boundary drift answers "did the active expert neighborhood change?" Transition drift answers "did the transition law itself change?" Both must be reported because top-k overlap can remain unchanged while router mass redistributes within the same active set.

Proposition 4.6 (Transition drift is not reducible to boundary drift)**.** *Boundary overlap and transition-law similarity are logically distinct. There exist probe windows for which the active top-k set is unchanged while router mass redistributes enough to change $\Delta^{\text{sem}}_{\text{transition}}$, and conversely probe windows for which the active set changes while the surviving mass remains concentrated on a common subset. Therefore both channel families are required for a complete router-preservation claim.*

4.5 Chart-content drift

Chart-content drift compares checkpoint-native expert fields on the same anchored conceptual point. The evaluation protocol is local to the instantiated base atlas rather than exhaustive over all experts.

Definition 4.7 (Checkpoint-native anchored expert field). *Fix a probe token t, protected layer ℓ, checkpoint θ_j, and expert $e \in \mathcal{E}_{\mathrm{loc}}(t, \ell)$. Let*

$$\iota^{(\theta_j)}_{e,t,\ell}(z_{0,t,\ell}) \tag{51}$$

denote the exact tensor that checkpoint θ_j would feed to expert e when the layer is anchored on the base canonical state $z_{0,t,\ell}$ and passed through checkpoint θ_j's native pre-expert path. Let

$$u^{(\theta_j)}_{e,t,\ell} := \mathrm{Expert}_{\theta_j,e}\big(\iota^{(\theta_j)}_{e,t,\ell}(z_{0,t,\ell})\big) \tag{52}$$

denote the resulting raw expert update in residual coordinates. Let $P_{0,t,\ell}$ denote the base-anchor first-order visible projector induced by $z_{0,t,\ell}$. Define the anchored tangent-visible expert field

$$\widetilde{u}^{(\theta_j)}_{e,t,\ell} := P_{0,t,\ell} u^{(\theta_j)}_{e,t,\ell}. \tag{53}$$

Definition 4.8 (Chart-content drift). *Fix a pre-registered content quantile $q_{\mathrm{content}} \in [0.5, 1]$. For a protected layer ℓ, define*

$$\Delta_{\mathrm{content}}(\ell) := \mathbb{E}_t \, \mathrm{Quantile}_{q_{\mathrm{content}}; e \in \mathcal{E}_{\mathrm{loc}}(t,\ell)} \left[\frac{\left\| \widetilde{u}^{(\theta_1)}_{e,t,\ell} - \widetilde{u}^{(\theta_0)}_{e,t,\ell} \right\|_2}{\left\| \widetilde{u}^{(\theta_1)}_{e,t,\ell} \right\|_2 + \left\| \widetilde{u}^{(\theta_0)}_{e,t,\ell} \right\|_2 + \epsilon} \right]. \tag{54}$$

Interpretation. The local expert set keeps the measurement on-manifold relative to the base atlas actually used by the model. Choosing $q_{\mathrm{content}} > 0.5$ makes the functional explicitly sensitive to high-drift tails rather than only to the median expert behavior.

4.6 Sufficient statistics for independent re-bootstrap

All confidence intervals in this specification are bootstrap intervals over windows. The stored artifacts must therefore be sufficient to recompute every window-level statistic without access to the original model run.

Definition 4.9 (Per-window sufficient statistics). *Fix a protected layer ℓ and a window w. For each metric channel below, define the per-window mean over valid probe tokens in w:*

$$\delta^{z}_{\mathrm{coord}}(w,\ell), \quad \delta^{g}_{\mathrm{coord}}(w,\ell), \quad \delta^{\mathrm{sem}}_{\mathrm{boundary}}(w,\ell), \tag{55}$$

$$\delta^{\mathrm{router}}_{\mathrm{boundary}}(w,\ell), \quad \delta^{\mathrm{gain}}_{\mathrm{boundary}}(w,\ell), \quad \delta^{\mathrm{sem}}_{\mathrm{transition}}(w,\ell), \tag{56}$$

$$\delta^{\mathrm{router}}_{\mathrm{transition}}(w,\ell), \quad \delta^{\mathrm{gain}}_{\mathrm{transition}}(w,\ell), \quad \delta_{\mathrm{content}}(w,\ell). \tag{57}$$

The artifact must store, for every (w, ℓ) and every channel:

1. *the per-window mean value;*
2. *the valid-token count used to form that mean;*
3. *for chart-content drift, the declared q_{content} and the local expert-set mode used to define $\mathcal{E}_{\mathrm{loc}}(t, \ell)$.*

These statistics are sufficient to reconstruct every layerwise bootstrap CI in this section under the recorded bootstrap protocol.

Proposition 4.10 (Per-window artifacts are sufficient for independent re-bootstrap)**.** *Fix the bootstrap protocol of Definition 5.1. Given the per-window channel means, valid-token counts, and content-metadata fields in Definition 4.9, an independent verifier can reconstruct every layerwise canonical and mandatory-diagnostic bootstrap interval in this section without re-running model inference.*

Requirement 4.11 (Canonical and diagnostic channels are both mandatory)**.** Any artifact that reports boundary or transition drift must report all three channels: semantic, router-only, and gain-only. Any artifact that reports coordinate drift must report both the pre-gain canonical channel and the post-gain gate channel. Reporting only the collapsed canonical value is insufficient.

5 Failure Predicates and Verification Requirements

This section turns the drift functionals into executable verification predicates. The main separation is between:

1. canonical preservation predicates, which decide whether the protected atlas has been rewritten;
2. output-damage predicates, which decide whether that rewrite produced base-domain harm;
3. task-classification predicates, which decide whether an objective is ordinary post-training or knowledge-expansive.

5.1 Bootstrap and confidence-interval semantics

Definition 5.1 (Bootstrap protocol)**.** *All confidence intervals in this section are non-parametric bootstrap intervals over windows. The protocol is:*

1. *the resampling unit is the window, not the token;*
2. *the number of bootstrap replicates is pre-registered and recorded as $B \geq 200$;*
3. *the default confidence interval is the percentile interval at level $1 - \alpha$ with $\alpha = 0.05$, unless a stricter interval family is explicitly pre-registered;*
4. *the bootstrap seed is recorded in the artifact;*
5. *every bootstrap statistic is recomputed from the per-window sufficient statistics in Definition 4.9, not from already-aggregated global means.*

Canonical-vs-diagnostic rule. Unless explicitly stated otherwise, failure predicates are evaluated on the canonical channels: $\Delta^{z}_{\text{coord}}$, $\Delta^{g}_{\text{coord}}$, $\Delta^{\text{sem}}_{\text{boundary}}$, $\Delta^{\text{sem}}_{\text{transition}}$, and Δ_{content}. Router-only and gain-only quantities are mandatory diagnostics used for attribution, not for replacing the canonical predicate.

Proposition 5.2 (Failure evaluation is canonical-channel first)**.** *For F1, the truth value of the predicate is determined entirely by the canonical channels together with the pre-registered thresholds and bootstrap protocol. Mandatory diagnostic channels refine attribution after F1 fires, but they do not change the truth value of F1 itself.*

5.2 Failure predicates

Definition 5.3 (Protected-atlas rewrite thresholds). *Fix pre-registered thresholds*

$$\tau^{z}_{\text{coord}}, \quad \tau^{g}_{\text{coord}}, \quad \tau_{\text{content}}, \quad \tau^{\text{sem}}_{\text{boundary}}, \quad \tau^{\text{sem}}_{\text{transition}}. \tag{58}$$

These thresholds are part of the verification manifest and must be declared before the comparison run is evaluated.

Definition 5.4 (Failure Predicate F1: protected-atlas rewrite). *Fix a base checkpoint θ_0, a comparison checkpoint θ_1, a probe family $\mathcal{P}$, a protected layer set $\mathcal{L}_{\text{protect}}$, and thresholds as in Definition 5.3. Declare* protected-atlas rewrite *if there exists $\ell \in \mathcal{L}_{\text{protect}}$ such that at least one of the following holds:*

$$\inf \mathrm{CI}_{1-\alpha}\big(\Delta^{z}_{\text{coord}}(\ell)\big) > \tau^{z}_{\text{coord}}, \tag{59}$$
$$\inf \mathrm{CI}_{1-\alpha}\big(\Delta^{g}_{\text{coord}}(\ell)\big) > \tau^{g}_{\text{coord}}, \tag{60}$$
$$\inf \mathrm{CI}_{1-\alpha}\big(\Delta_{\text{content}}(\ell)\big) > \tau_{\text{content}}, \tag{61}$$
$$\inf \mathrm{CI}_{1-\alpha}\big(\Delta^{\text{sem}}_{\text{boundary}}(\ell)\big) > \tau^{\text{sem}}_{\text{boundary}}, \tag{62}$$
$$\inf \mathrm{CI}_{1-\alpha}\big(\Delta^{\text{sem}}_{\text{transition}}(\ell)\big) > \tau^{\text{sem}}_{\text{transition}}. \tag{63}$$

Attribution after F1 fires. If F1 holds, the artifact must report the corresponding router-only and gain-only channels at the same layer so that the rewrite can be attributed to transport drift, gain-interface drift, router-core drift, chart-content drift, or any combination thereof.

Definition 5.5 (Failure Predicate F2: base-domain damage). *Fix a pre-registered canary set $\mathcal{C}_{\text{base}}$ of base-domain evaluation windows and a pre-registered output-metric family $\mathfrak{M}_{\text{base}}$. Typical canonical metrics include base-domain loss regression, KL drift from base, and calibration error. Entropy-collapse diagnostics may be reported as adjacent diagnostics, but do not trigger F2 by themselves unless a later specification explicitly promotes them to canonical status. Declare* base-domain damage *if at least one metric $M \in \mathfrak{M}_{\text{base}}$ shows CI-separated degradation on $\mathcal{C}_{\text{base}}$ relative to the base checkpoint under its pre-registered sign convention.*

Definition 5.6 (Failure Predicate F3: knowledge-expansive classification). *Fix a task objective with success metric S, target level $S_\star$, a probe family $\mathcal{P}$, a protected layer set $\mathcal{L}_{\text{protect}}$, and a pre-registered family $\mathfrak{P}_{\text{cp}}$ of chart-preserving policies. Declare the task* knowledge-expansive *if:*

1. *every policy $\pi \in \mathfrak{P}_{\text{cp}}$ fails to reach $S_\star$ under the matched-success protocol, and*
2. *there exists a broader policy class permitting chart insertion, continued pretraining, or chart-redefining adaptation that does reach $S_\star$.*

5.3 Verification requirements

Requirement 5.7 (Matched-success discipline). No preservation claim is valid unless compared at matched task success. A method does not earn credit for preserving the atlas by underperforming the task objective. Where exact matching is impossible, the artifact must report the full success frontier together with the interpolation rule used for comparison.

Requirement 5.8 (Fixed probes, canaries, and protected layers). Every verification artifact must name:

1. the probe family $\mathcal{P}$ used for atlas drift;
2. the protected layer set $\mathcal{L}_{\text{protect}}$;
3. the base canary set $\mathcal{C}_{\text{base}}$ used for F2;
4. the threshold manifest used for F1.

Without all four, the result is not auditable.

Requirement 5.9 (Router-first accounting)**.** Any claim about chart preservation must report both boundary and transition drift, together with their router-only and gain-only decompositions. Regularizing expert fields while omitting transition-map reinterpretation is not sufficient.

Requirement 5.10 (Surface ordering)**.** Interventions intended to remain chart-preserving should escalate in the following order unless a deliberate exception is pre-registered:

1. readout and small adapters;
2. endpoint and low-risk surfaces;
3. bounded router-prior or dispatch-surface control;
4. chart insertion or continued pretraining;
5. only then broad protected-interior chart motion.

The artifact must declare the highest intervention surface activated by the run.

Requirement 5.11 (No silent success claims under rewrite)**.** If a run improves task success while triggering F1 on protected layers, the result must be labeled *chart-redefining.* Any success claim that omits this label is incomplete.

Requirement 5.12 (Failure attribution is mandatory)**.** Whenever F1 holds at a protected layer, the verification report must include the corresponding canonical and diagnostic drift channels, so the rewrite can be attributed to concept motion, gain-interface drift, router-core drift, content drift, or mixed causes.

5.4 Decision logic

Definition 5.13 (Chart-preserving pass)**.** *A post-training run is* chart-preserving *on* $(\mathcal{P}, \mathcal{L}_{\text{protect}})$ *if and only if:*

1. *F1 does not hold;*
2. *the run satisfies the matched-success protocol in Requirement 5.7;*
3. *the artifact satisfies Requirements 5.8–5.12.*

Definition 5.14 (Chart-preserving with damage, chart-redefining, and knowledge-expansive outcomes)**.** *Given a run or task family:*

1. *if F1 is false and F2 is false, the run is* chart-preserving without base-domain damage*;*
2. *if F1 is false and F2 is true, the run is* chart-preserving but output-damaging*;*
3. *if F1 is true, the run is* chart-redefining*;*
4. *if F3 is true for the objective class, the task is* knowledge-expansive *relative to the declared chart-preserving policy family.*

Interpretation. F1 answers whether the protected atlas was rewritten. F2 answers whether the user-visible base-domain surface was damaged. F3 answers whether the objective itself belongs in ordinary post-training or in a broader knowledge-expansion regime. Together they define the minimum receipt required for a credible atlas-preservation claim.

Proposition 5.15 (Run-level outcome partition)**.** *Fix a matched-success run with a valid artifact bundle. Exactly one of the following run-level outcomes holds:*

1. *chart-preserving without base-domain damage;*
2. *chart-preserving with base-domain damage;*
3. *chart-redefining.*

Knowledge-expansive classification is an objective-level overlay determined by F3 on a policy family rather than an additional run-level state.

6 Artifact Families and Schema Contract

This section defines the artifact contract for atlas-verification claims. The contract is intentionally narrower than "store everything": it requires enough information to re-bootstrap every canonical drift functional and every failure predicate, while distinguishing canonical pass/fail objects from mandatory diagnostics and adjacent diagnostics.

6.1 Design principles

Window-bootstrap sufficiency. All canonical confidence intervals in this specification are computed by non-parametric bootstrap over windows. Accordingly, every canonical artifact must store per-window sufficient statistics for the pre-registered functionals it reports. Token-level replay is optional; window-level re-bootstrap is mandatory.

Canonical versus diagnostic quantities. The artifact contract uses three roles:

1. **Canonical:** quantities that directly determine the drift functionals, confidence intervals, and failure predicates.
2. **Mandatory diagnostic:** quantities that do not themselves define pass/fail, but must be reported alongside a canonical quantity because they disambiguate failure mode.
3. **Adjacent diagnostic:** quantities that are scientifically useful but are not part of the canonical atlas-drift contract.

For example, semantic boundary drift is canonical, while the router-only and gain-only decompositions are mandatory diagnostics. Boundary amplification and terminal tracking are adjacent diagnostics.

No hidden recomputation. A produced artifact bundle is acceptable only if an independent verifier can reproduce every reported CI and every failure label from the stored arrays and the recorded bootstrap settings, without re-running model inference.

Proposition 6.1 (Bundles are evidentiary objects, not storage conveniences)**.** *In this specification, an artifact bundle is the minimal evidentiary object for an atlas-preservation claim. Schema validity, linkage validity, and re-bootstrap sufficiency are therefore part of the claim itself rather than merely implementation hygiene.*

6.2 Artifact families and recommended layout

Definition 6.2 (Canonical checkpoint-comparison bundle)**.** *The minimal canonical bundle for a checkpoint-comparison claim consists of:*

1. *`manifest.json`;*
2. *`probe_spec.json`;*
3. *`atlas_drift.json`;*
4. *`bootstrap_report.json`;*
5. *`failure_report.json`.*

If the claim includes output-damage or matched-success conclusions, the bundle must additionally include `output_damage.json`. If the claim includes F3 or matched-success comparison across policies or runs, the bundle must additionally include `comparison_bundle.json`.

An optional `checkpoint_extract.jsonl` artifact may additionally be emitted for raw auditability of the single-checkpoint forward pass, but it is not required for window-bootstrap reproducibility.

Requirement 6.3 (Canonical bundle completeness)**.** No canonical claim is valid unless every artifact required by Definition 6.2 for that claim type is present, schema-valid, and linked from the manifest. Optional artifacts may strengthen auditability, but they cannot substitute for a missing required artifact.

A recommended layout is:

```
OUT_ROOT/
  <run_id>/
    manifest.json
    probe_spec.json
    atlas_drift.json
    bootstrap_report.json
    failure_report.json
    output_damage.json              # required for F2 / matched-success claims
    comparison_bundle.json          # required for F3 / matched-success claims
    checkpoint_extract_theta0.jsonl # optional raw extraction audit trail
    checkpoint_extract_theta1.jsonl # optional raw extraction audit trail
    legacy_support/                 # optional absorbed legacy artifacts
      boundary_amplification.json
      chart_compatibility.json
      terminal_tracking.json
      kl_edge_dominance.json
```

6.3 Common metadata contract

Every artifact family must include the following common fields:

1. **Identity:** `schema_version`, `artifact_family`, `artifact_role`, `run_id`.

2. **Checkpoint provenance:** base checkpoint id, comparison checkpoint id when applicable, and exact model family / gate-semantics profile identifier.
3. **Data provenance:** tokenizer identifier, window sampler, `n_windows`, `seq_len`, text-field semantics when applicable, and deterministic seeds.
4. **Bootstrap provenance:** resampling unit, replicate count, CI type, and bootstrap seed.
5. **Linkage:** paths or content hashes for the other artifacts in the same run bundle.

The minimal manifest schema is:

```
{
  "schema_version": 1,
  "artifact_family": "manifest",
  "artifact_role": "canonical",
  "run_id": "atlas_cmp_2026_03_06_001",
  "paper": "atlas_verification_spec",
  "git": {"commit": "...", "dirty": true},
  "checkpoints": {"base": "...", "comparison": "..."},
  "links": {
    "probe_spec": "probe_spec.json",
    "atlas_drift": "atlas_drift.json",
    "bootstrap_report": "bootstrap_report.json",
    "failure_report": "failure_report.json",
    "output_damage": "output_damage.json",
    "comparison_bundle": "comparison_bundle.json"
  },
  "created_utc": "...",
  "producer": {"script": "...", "hostname": "..."}
}
```

Requirement 6.4 (Manifest linkage is normative)**.** The manifest is the authoritative linkage object for the bundle. If a canonical artifact is present on disk but absent from the manifest, or linked from the manifest but missing on disk, the bundle is invalid.

6.4 Probe specification artifact

Purpose. `probe_spec.json` is the measurement pre-registration for a run. It defines the probe family, protected layers, gate semantics, expert-evaluation set semantics, and any thresholds needed to interpret later failure reports.

Canonical fields. The artifact must record:

1. the probe family $\mathcal{P}$: window sampler, tokenizer semantics, deterministic seed, and sequence length;
2. the protected layer set $\mathcal{L}_{\text{protect}}$;
3. the gate-semantics profile, including exact top-k, masks, tie-breaking, score dtype, and shared-expert exclusion;

4. the canonical margin semantics

$$m_{\text{set}} := s_{(k)} - s_{(k+1)},$$

   computed on the admissible routed expert set after deterministic masks and before any softmax or weight renormalization;

5. the canonical expert evaluation set semantics: base top-k union $\{e_{k+1}\}$;

6. the pre-registered content quantile q_{content};

7. any decision thresholds later used by F1–F3, if those thresholds are fixed at measurement time rather than bundle time.

A minimal schema is:

```
{
  "schema_version": 1,
  "artifact_family": "probe_spec",
  "artifact_role": "canonical",
  "probe_family": {
    "tokenizer": "...",
    "window_sampler": "...",
    "n_windows": 200,
    "seq_len": 256,
    "seed": 12345
  },
  "protected_layers": [30, 40, 50],
  "gate_semantics": {
    "profile": "...",
    "topk": 8,
    "scores_dtype": "fp32",
    "shared_expert_excluded": true,
    "masks": "post-mask admissible routed expert set",
    "m_set_definition": "raw_dispatch_gap_post_mask_pre_softmax"
  },
  "expert_eval_set": {
    "canonical": "base_topk_plus_kplus1",
    "exhaustive_label": "noncanonical_exhaustive"
  },
  "content_quantile": 0.75,
  "thresholds": {
    "coord_pre": 0.0,
    "coord_gate": 0.0,
    "content": 0.0,
    "boundary": 0.0,
    "transition": 0.0
  }
}
```

6.5 Canonical atlas-drift artifact

Purpose. `atlas_drift.json` is the canonical comparison artifact. It stores the per-window sufficient statistics required to re-bootstrap every atlas-drift CI in the specification. It does *not* need to store token-level hidden states if the registered functionals can be reproduced from stored per-window arrays.

Canonical arrays. For each protected layer ℓ, the artifact must store per-window arrays for:

1. pre-gain coordinate drift;
2. post-gain gate coordinate drift;
3. semantic boundary drift;
4. semantic transition drift;
5. chart-content drift.

Those arrays are the canonical inputs to F1.

Mandatory diagnostics. For each protected layer ℓ, the artifact must also store per-window arrays for:

1. router-only boundary drift;
2. gain-only boundary drift;
3. router-only transition drift;
4. gain-only transition drift.

These do not replace the canonical semantic quantities. They must be reported because they disambiguate whether a failed semantic comparison comes primarily from router-core drift, gain-interface drift, or both.

Optional adjacent diagnostics. The artifact may additionally store per-window arrays for exhaustive-expert content drift or off-manifold evaluations, but those must be explicitly labeled non-canonical.

A minimal schema is:

```
{
  "schema_version": 1,
  "artifact_family": "atlas_drift",
  "artifact_role": "canonical",
  "layers": [30, 40, 50],
  "per_window": {
    "delta_coord_pre": {"30": [..], "40": [..], "50": [..]},
    "delta_coord_gate": {"30": [..], "40": [..], "50": [..]},
    "delta_boundary_sem": {"30": [..], "40": [..], "50": [..]},
    "delta_boundary_router_only": {"30": [..], "40": [..], "50": [..]},
    "delta_boundary_gain_only": {"30": [..], "40": [..], "50": [..]},
    "delta_transition_sem": {"30": [..], "40": [..], "50": [..]},
    "delta_transition_router_only": {"30": [..], "40": [..], "50": [..]},
    "delta_transition_gain_only": {"30": [..], "40": [..], "50": [..]},
    "delta_content": {"30": [..], "40": [..], "50": [..]}
  },
  "semantics": {
    "boundary": "canonical_semantic_base_anchor_checkpoint_native_gain",
    "transition": "canonical_semantic_base_anchor_checkpoint_native_gain",
    "content": "canonical_local_base_topk_plus_kplus1"
  },
  "diagnostics": {
    "router_only": true,
    "gain_only": true,
    "exhaustive_content": false
  }
}
```

Sufficiency requirement. If a canonical drift functional is reported in `bootstrap_report.json` but the corresponding per-window array is absent from `atlas_drift.json`, the artifact bundle is invalid.

Proposition 6.5 (Summary artifacts are not self-sufficient). *`bootstrap_report.json` is a canonical summary artifact but not a self-sufficient evidentiary artifact. Without the corresponding per-window arrays in `atlas_drift.json` and, when relevant, `output_damage.json`, an independent verifier cannot reconstruct the reported confidence intervals and therefore cannot validate the claim.*

6.6 Optional checkpoint-extraction artifact

Purpose. `checkpoint_extract_theta*.jsonl` is an optional raw-audit artifact that records single-checkpoint forward-pass extraction results before cross-checkpoint aggregation. It is useful when the implementation of the observable extraction contract itself is under audit. It is not required for window-bootstrap reproducibility if `atlas_drift.json` is complete.

Recommended fields. Each line should correspond to a window and may store compressed per-layer summaries of:

1. canonical pre-gain coordinate norms or hashes;
2. post-gain gate coordinate norms or hashes;
3. top-k routed sets and e_{k+1} identities;
4. set margins;
5. router outputs on checkpoint-native coordinates.

Because this artifact is optional and storage-heavy, the specification does not require a single canonical serialization beyond the common metadata contract.

6.7 Bootstrap report artifact

Purpose. `bootstrap_report.json` stores the confidence intervals and bootstrap settings derived from the per-window arrays. It is the canonical statistical summary layer, but it is not sufficient by itself because it does not permit independent re-bootstrap unless paired with `atlas_drift.json` and, where relevant, `output_damage.json`.

A minimal schema is:

```
{
  "schema_version": 1,
  "artifact_family": "bootstrap_report",
  "artifact_role": "canonical",
  "bootstrap": {
    "resampling_unit": "window",
    "B": 200,
    "ci_type": "percentile",
    "seed": 12345
  },
  "summary": {
    "delta_coord_pre": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}},
    "delta_coord_gate": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}},
    "delta_boundary_sem": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}},
    "delta_boundary_router_only": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}},
    "delta_boundary_gain_only": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}},
    "delta_transition_sem": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}},
    "delta_transition_router_only": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}},
    "delta_transition_gain_only": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}},
    "delta_content": {"30": {"mean": 0.0, "ci95": [0.0, 0.0]}}
  }
}
```

6.8 Output-damage artifact

Purpose. `output_damage.json` stores the per-window sufficient statistics for output-level damage variables. It is required whenever F2, matched-success, or output-damage claims are made. It is not itself an atlas-drift artifact, but it is the canonical evidence layer for user-visible damage.

Canonical output-damage arrays. Depending on the registered claim, the artifact may store per-window arrays for:

1. base-domain loss regression;
2. KL drift from base;
3. calibration damage;
4. entropy or varentropy diagnostics.

Only the first three are candidates for canonical F2 use. Entropy-style metrics are adjacent diagnostics unless a later spec explicitly promotes them.

A minimal schema is:

```
{
  "schema_version": 1,
  "artifact_family": "output_damage",
  "artifact_role": "canonical",
  "per_window": {
    "loss_regression": [..],
    "kl_from_base": [..],
    "calibration_damage": [..],
    "entropy_mean": [..],
    "varentropy_mean": [..]
  },
  "diagnostics": {
    "entropy_mean": "adjacent_diagnostic",
    "varentropy_mean": "adjacent_diagnostic"
  }
}
```

6.9 Failure-report artifact

Purpose. `failure_report.json` is the decision layer. It records the thresholds, the CI comparisons, and the resulting F1–F3 labels. It is invalid unless the corresponding canonical arrays exist in `atlas_drift.json` and, when relevant, `output_damage.json`.

A minimal schema is:

```
{
  "schema_version": 1,
  "artifact_family": "failure_report",
  "artifact_role": "canonical",
  "thresholds": {
    "coord_pre": 0.0,
    "coord_gate": 0.0,
    "content": 0.0,
    "boundary": 0.0,
    "transition": 0.0,
    "output_damage": 0.0
  },
  "F1": {
    "protected_atlas_rewrite": true,
    "per_layer": {
      "30": {
        "coord_gate_ci95": [0.0, 0.0],
        "content_ci95": [0.0, 0.0],
        "boundary_sem_ci95": [0.0, 0.0],
        "transition_sem_ci95": [0.0, 0.0],
        "triggered": true
      }
    }
  },
  "F2": {
    "base_domain_damage": false,
    "metrics": {
      "loss_regression_ci95": [0.0, 0.0],
      "kl_from_base_ci95": [0.0, 0.0],
      "calibration_damage_ci95": [0.0, 0.0]
    }
  },
  "F3": {
    "knowledge_expansive": false,
    "status": "not_evaluated_or_bundle_required"
  },
  "classification": "chart_preserving"
}
```

F3 and bundles. Because F3 is inherently comparative across policies or checkpoints, a single run-level failure report may record F3 as unevaluated. A valid positive F3 claim requires a comparison bundle.

6.10 Comparison-bundle artifact

Purpose. `comparison_bundle.json` aggregates multiple run bundles for matched-success comparisons, policy-family comparisons, or knowledge-expansive classification. It is the only valid place to make F3 claims or matched-success superiority claims.

Canonical bundle fields. A valid comparison bundle must record:

1. the participating runs and their linked `failure_report.json` artifacts;
2. the success metric and matched-success tolerance;
3. the decision rule for declaring a policy family chart-preserving or chart-redefining;
4. any absorbed legacy artifacts, labeled as `legacy_support` rather than canonical evidence.

A minimal schema is:

```
{
  "schema_version": 1,
  "artifact_family": "comparison_bundle",
  "artifact_role": "canonical",
  "success_spec": {
    "metric": "...",
    "target": 0.0,
    "match_tolerance": 0.0
  },
  "runs": [
    {"run_id": "...", "policy_family": "chart_preserving", "failure_report": "..."},
    {"run_id": "...", "policy_family": "unconstrained", "failure_report": "..."}
  ],
  "legacy_support": {
    "boundary_amplification": ["legacy_support/boundary_amplification.json"],
    "chart_compatibility": ["legacy_support/chart_compatibility.json"]
  },
  "summary": {
    "matched_success": true,
    "F3": {"knowledge_expansive": false},
    "winner": "chart_preserving"
  }
}
```

6.11 Validity rules

The following rules are mandatory:

1. A canonical claim about atlas drift is invalid without `manifest.json`, `probe_spec.json`, `atlas_drift.json`, and `bootstrap_report.json`.
2. A canonical F1 label is invalid unless every reported triggering metric is backed by a stored per-window array.
3. A canonical F2 label is invalid without `output_damage.json`.
4. A canonical F3 label is invalid without `comparison_bundle.json`.
5. Any legacy artifact included in the run must be labeled `legacy_support` and may not be used to satisfy the canonical validity conditions above.

These rules make the ontology operational. The canonical bundle is the sole path from claimed atlas preservation or rewrite to independently checkable receipt.

Proposition 6.6 (Schema validity is necessary but not sufficient)**.** *Passing every per-file schema is necessary for bundle validity but not sufficient. Canonical validity additionally requires cross-file agreement on linkage, protected layers, thresholds, bootstrap semantics, claim-type-dependent artifact presence, and compliance with the legacy non-substitution rules in Section 7.*

7 Legacy Mapping and Absorption Policy

The routing-geometry artifacts already in the repository are useful prior work, but they are not the canonical atlas-verification contract. This section defines how they are absorbed. The principle is simple: old artifacts may contribute supporting evidence, reusable primitives, or diagnostic context, but they do not replace the canonical atlas-drift bundle defined in Section 6.

7.1 Mapping classes

We distinguish three classes of relationship between an older routing-geometry artifact and the new atlas-verification contract.

Definition 7.1 (Exact mapping)**.** *An old artifact or field maps exactly only when it instantiates the same mathematical object, under the same gate semantics, bootstrap semantics, and admissibility rules, as the new specification. Exact mappings may be imported directly into canonical metadata or comparison metadata.*

Definition 7.2 (Partial mapping)**.** *An old artifact partially maps when it measures a scientifically adjacent object that constrains or supports a canonical claim, but does not instantiate the canonical object itself. Partial mappings may be imported as* `legacy_support`*, but they cannot satisfy canonical validity requirements.*

Definition 7.3 (Non-canonical diagnostic)**.** *An old artifact is non-canonical when it is useful for interpretation or exploratory diagnosis but has no direct pass/fail role in the new contract. These artifacts may still be bundled and discussed, but they carry no canonical evidentiary weight for F1–F3.*

7.2 Primitive subobjects that carry over exactly

Not all legacy content is merely partial. Several primitive measurement choices carry over unchanged and may be imported exactly into the new probe and manifest layers:

1. fp32 score computation for routed expert ranking;
2. exact top-k routed set semantics;
3. the identities of the k-th and $(k+1)$-th competitors;
4. the ranking-margin object
$$m_{\text{set}} = s_{(k)} - s_{(k+1)},$$
computed post mask and pre softmax on the admissible routed expert set;
5. bootstrap over windows rather than tokens;
6. explicit required-layer declarations in artifacts.

These are exact subobject mappings. They should be preserved where they appear in the new artifact families. The important point is that exact *subobjects* do not imply exact test-level mapping. Most legacy tests still measure only a slice of the new ontology.

Proposition 7.4 (Exact subobjects do not promote a legacy test to canonical status)**.** *If a legacy artifact shares one or more exact primitive subobjects with the new contract, that suffices only for exact import of those subobjects. It does not imply that the legacy artifact as a whole instantiates the same comparison object, failure predicate, or evidentiary role as the canonical atlas-verification bundle.*

7.3 Test-level mapping table

Legacy artifact	Mapping class	Status in the new contract
Boundary amplification	Partial mapping	Measures low-margin concentration of cross-checkpoint routing disagreement on shared inputs. Useful evidence that boundary-locality matters, but it does not implement canonical semantic / router-only / gain-only boundary drift and therefore cannot replace Δ_{boundary}.
Chart compatibility	Partial mapping	Measures within-checkpoint adjacent-chart redundancy using e_k vs e_{k+1} plus random-e2 controls. Useful precursor to local chart-content reasoning, but it is not cross-checkpoint Δ_{content} and does not measure atlas rewrite directly.
Terminal ov_k scan / terminal tracking	Non-canonical diagnostic	Diagnoses terminal-interface regime and may inform layer-selection policy or endpoint interpretation. It does not measure protected mid-band atlas preservation and therefore carries no canonical F1 weight.
KL edge dominance / KL-bounded adaptation	Partial mapping	Lives on the output-damage or intervention side rather than inside atlas drift itself. It can support F2 or matched-success interpretation, but cannot substitute for canonical atlas-drift artifacts.
Behavioral compression	Non-canonical diagnostic	Compares behavioral change under routing-margin stratification. Useful for narrative interpretation, but not part of the canonical atlas-drift receipt path.
Entropy collapse	Non-canonical diagnostic	Useful as an output-shape diagnostic. It is not a canonical atlas-drift object and cannot trigger F1–F3 by itself.

7.4 Absorption rules by legacy artifact

Boundary amplification. The old boundary-amplification artifact should be absorbed as evidence about *boundary locality*, not as the canonical boundary-drift object. Its strongest contribution is to support the proposition that disagreement mass concentrates near the ranking boundary. What it lacks is the new cross-checkpoint semantic decomposition:

1. it does not separate semantic, router-only, and gain-only boundary drift;
2. it is stratified by legacy edge direction rather than by the full base-anchor semantic alignment contract;
3. it does not directly participate in F1.

Accordingly, a boundary-amplification artifact may appear under `legacy_support.boundary_amplification`, but it cannot satisfy the requirement for `atlas_drift.json`.

Chart compatibility. The old chart-compatibility artifact is the closest precursor to local chart-content reasoning, because it already uses adjacent competitors and the set margin. However, it remains a within-checkpoint redundancy test. It does not compare θ_0 and θ_1 on a common anchored state and therefore cannot stand in for Δ_{content}. Its correct role is:

1. support the claim that adjacency-specific redundancy exists in the base atlas;
2. justify why the canonical expert evaluation set remains local to base top-$k \cup \{e_{k+1}\}$;
3. provide legacy context when interpreting a later content-drift failure.

It is therefore a partial mapping, not an exact one.

Terminal tracking. The old terminal-scan and terminal-tracking artifacts should be treated as endpoint diagnostics. They are useful for understanding terminal-interface regimes and for deciding whether a layer belongs in the protected mid-band set. They are not preservation artifacts. In particular, a model may exhibit a stable terminal cliff and still rewrite the protected atlas interior, or preserve the protected atlas while changing terminal-interface diagnostics. The terminal artifacts therefore remain non-canonical diagnostics.

KL-bounded adaptation and KL edge dominance. The KL-family artifacts are best understood as output-damage and intervention-side evidence. They may help show that a policy family preserves useful behavior or that broader tuning induces more base-domain regression. They do not define atlas rewrite. The new contract therefore treats them as follows:

1. if tied to fixed probes and bundled with canonical atlas-drift artifacts, they may support F2 or matched-success interpretation;
2. if reported alone, they are insufficient to claim chart preservation;
3. if an intervention succeeds while F1 triggers, KL-side evidence may still be useful, but the run must be labeled chart-redefining.

7.5 Exact mappings, partial mappings, and prohibited substitutions

The following policy is normative.

Allowed exact import. Legacy fields that instantiate exact primitive subobjects—for example fp32 ranking margins, required-layer declarations, or window-bootstrap settings—may be copied into new canonical metadata without relabeling the mathematical object.

Allowed partial absorption. Legacy run artifacts that measure adjacent phenomena may be attached under `legacy_support` in a comparison bundle. They may strengthen interpretation, motivate hypotheses, or explain why a failure mode is plausible.

Prohibited substitution. The following substitutions are invalid:

1. using boundary amplification in place of canonical semantic boundary drift;
2. using chart compatibility in place of canonical chart-content drift;
3. using terminal tracking in place of protected-layer preservation evidence;
4. using KL-side artifacts in place of the canonical atlas-drift bundle.

Requirement 7.5 (Legacy non-substitution)**.** No artifact classified under Definitions 7.2 or 7.3 may discharge a canonical validity obligation that belongs to `atlas_drift.json`, `bootstrap_report.json`, `failure_report.json`, `output_damage.json`, or `comparison_bundle.json`.

7.6 Legacy-support bundle semantics

When a legacy artifact is carried forward, it must be explicitly labeled as legacy support. A recommended structure is:

```
{
  "legacy_support": {
    "boundary_amplification": ["legacy_support/boundary_amplification.json"],
    "chart_compatibility": ["legacy_support/chart_compatibility.json"],
    "terminal_tracking": ["legacy_support/terminal_tracking.json"],
    "kl_edge_dominance": ["legacy_support/kl_edge_dominance.json"]
  }
}
```

This labeling matters because the new contract is intentionally asymmetric: legacy artifacts may explain a canonical result, but they may never be used to manufacture one.

7.7 Practical consequence

The older routing-geometry program is therefore not discarded. It is absorbed into a sharper ontology:

1. exact primitive routing objects are retained unchanged where valid;
2. legacy tests become supporting or diagnostic evidence;
3. the canonical receipt path for atlas preservation or rewrite runs only through the new artifact families.

That is the correct relationship between the two programs. The old artifacts remain scientifically useful, but the new spec is the source of truth for what counts as an auditable atlas-verification claim.

References

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems 32*, 2019.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. doi: 10.1016/j.neunet.2019.01.012.

A Notation and Naming Conventions

A.1 Notation summary

Symbol	Meaning
θ_0, θ_1	Base checkpoint and comparison checkpoint.
$\mathcal{P}$	Probe family: fixed window sampler, tokenizer semantics, and deterministic seeds.
$\mathcal{L}_{\text{protect}}$	Protected routed layer set used for atlas-preservation claims.
$z_{t,\ell}^{(\theta)}$	Canonical pre-gain operational state for token t at layer ℓ under checkpoint θ.
$g_{t,\ell}^{(\theta)}$	Exact post-gain gate coordinate under checkpoint θ.
$w_{t,\ell}^{(\theta)}$	Router output distribution or sparse routing weights under checkpoint θ.
$R_k^{(\theta)}(\cdot)$	Unordered top-k routed expert set under checkpoint θ.
$m_{\text{set}}^{(\theta)}$	Raw dispatch-score gap $s_{(k)} - s_{(k+1)}$ on the admissible routed expert set, post mask and pre softmax.
$\widetilde{u}_{e,t,\ell}^{(\theta)}$	Tangent-visible expert field for expert e evaluated under checkpoint θ.
$g_{0\to 0}$	Base-anchor gate coordinate obtained by applying $\Gamma^{(\theta_0)}$ to the base canonical state z_0.
$g_{0\to 1}$	Semantic cross-checkpoint gate coordinate obtained by applying $\Gamma^{(\theta_1)}$ to the same base canonical state z_0.
Δ_{coord}	Coordinate drift family, split into pre-gain canonical and post-gain gate forms.
Δ_{boundary}	Boundary drift family, with canonical semantic form and mandatory router-only / gain-only diagnostics.
$\Delta_{\text{transition}}$	Transition drift family, with canonical semantic form and mandatory router-only / gain-only diagnostics.
Δ_{content}	Chart-content drift on a pre-registered expert evaluation set.
$F1, F2, F3$	Protected-atlas rewrite, base-domain damage, and knowledge-expansive-task failure predicates.

A.2 Artifact naming conventions

The schema layer should keep names stable across implementations. The recommended convention is:

1. **Canonical quantities** use the atlas-language names in the paper, such as `coord_drift_zg`, `boundary_drift_semantic`, or `content_drift`.
2. **Diagnostic decompositions** are explicitly suffixed, such as `boundary_drift_router_only` and `transition_drift_gain_only`.
3. **Stored sufficient statistics** are named by resampling unit rather than by aggregate, such as `per_window.coord_gate` or `per_window.boundary_semantic`.
4. **Failure reports** should report both thresholds and confidence intervals, not only boolean pass/fail flags.

A.3 Traceability summary

Table 2: Claim-to-artifact traceability summary for the atlas verification contract.

Contract layer	Canonical object	Required artifact support	Expected verification output
Forward extraction	Single-checkpoint observables	Probe manifest + per-window extraction record	Reconstructable checkpoint-local atlas state
Alignment rules	Base-anchor comparison semantics	Comparison manifest + alignment metadata	Reconstructable semantic and diagnostic cross-checkpoint evaluations
Drift functionals	Coordinate, boundary, transition, content drift	Per-window sufficient statistics + bootstrap spec	Independently re-bootstrappable CI95 values
Failure predicates	F1, F2, F3	Threshold registry + failure report	Explicit rewrite / damage / objective-classification outcomes
Artifact schemas	Schema-valid manifests and bundles	Versioned JSON or equivalent artifact family	Auditable storage contract
Legacy absorption	Mapping from prior routing-geometry receipts	Bundle-level compatibility annotation	Clear distinction between exact, partial, and diagnostic legacy mappings

Use. The table is the compact claim-to-artifact traceability summary for the specification. Each row names the canonical contract layer, the object verified at that layer, the minimum artifact support required, and the expected verifier-visible output.