MoE Routing Attestation Specification

Document 14 — Auburn Governance Stack

Clause AI-MoE-1 — Auburn Patent Family

Fields

February 2026

UncleBroFields@proton.me
fieldsryanchristopher@gmail.com

Intellectual Property (IP) Declaration

The methods, logic structures, and attestation schemas contained in this document are the sole property of Ryan Fields.

Public License (Non-Commercial)
This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) license.

- **Academic Use:** Researchers may share and use this framework for non-commercial academic purposes, provided full attribution is given to Ryan Fields.
- **No Derivatives:** No modifications or "remixes" of the attestation schemas or normative requirements are permitted without express written consent.

Contact

UncleBroFields@proton.me
fieldsryanchristopher@gmail.com

Contents

1 Introduction

1.1 Purpose

This document specifies the attestation requirements for Mixture-of-Experts (MoE) routing decisions within the Auburn Governance Stack. As foundation models increasingly adopt MoE architectures to achieve parameter efficiency—activating a fraction of total parameters per token through learned routing functions—the routing decision itself becomes a critical audit surface that existing governance frameworks do not address.

A dense transformer applies every parameter to every token: the computational path is static and fully determined by the model weights. An MoE transformer *selects* which parameters to apply: the computational path is dynamic, token-dependent, and mediated by a learned gating function. This dynamic selection creates a governance gap. Without routing attestation, a verifier cannot determine which experts processed a given token, whether the routing was deterministic, whether any tokens were silently dropped, or whether the expert utilization distribution is consistent with the model's validated behavior.

This specification closes that gap by defining normative requirements for MoE routing determinism, routing decision logging, expert utilization monitoring, deployment topology disclosure, and load balancing state attestation. It extends the Model Attestation Interface (MAI-1) with routing-specific fields and defines three conformance levels (C0, C1, C2) that trade off attestation depth against operational cost.

1.2 Scope

This document addresses **inference-time** routing attestation only. Training-time routing behavior (including stochastic exploration mechanisms, auxiliary loss computation, and expert warm-up schedules) is outside the scope of this specification, except insofar as training-time mechanisms must be confirmed disabled during attested inference.

The architectures explicitly analyzed are:

- **DeepSeek-V3** (256 routed + 1 shared expert, sigmoid gating, top-8, auxiliary-loss-free bias balancing, node-limited routing).
- **Mixtral 8x7B** (8 experts, softmax gating, top-2, no capacity constraints).
- **Switch Transformer** (up to 2048 experts, softmax gating, top-1, capacity factor with token dropping).
- **ST-MoE** (32 experts, softmax gating with z-loss, capacity factor).

The normative requirements are architecture-agnostic; the architecture-specific analysis informs the requirement design and provides implementation context.

1.3 Relationship to MAI-1

This document supports all five mandatory invariants defined in MAI-1 (Document 5, Clause AI-5):

1. **Entropy Floor (AI-8):** For MoE models, the output entropy depends on which experts process the token. Routing non-determinism can cause entropy violations even when the model weights are unchanged.
2. **Gradient Starvation Envelope (AI-2):** Expert underutilization during training is the primary gradient starvation mechanism in MoE architectures. Routing utilization data provides the direct observable.

3. **Lyapunov Stability (AI-3):** Speculative decoding with MoE models requires routing consistency between draft and verification passes. Routing non-determinism can cause speculative tokens to be rejected at rates that exceed Lyapunov stability bounds.

4. **Distribution Drift (AI-6):** Expert utilization drift—a shift in which experts are selected over time—is a routing-specific manifestation of distribution drift that may occur even when the model's output distribution appears stable.

5. **Structural Coherence (AI-7):** Routing coherence—whether semantically similar tokens are routed to similar expert combinations—contributes to the model's structural coherence as measured by Dirichlet energy.

1.4 Normative Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 and RFC 8174.

1.5 Honest Framing

Honest Framing

This specification provides routing *reproducibility infrastructure*, not behavioral safety guarantees. Knowing which experts processed a token does not reveal whether the output is safe, truthful, or aligned with human values. What routing attestation provides is *auditability*: the ability to reconstruct the computational path after the fact, detect anomalous routing patterns, and verify that the routing mechanism operated as declared. The specification also acknowledges a fundamental physical limitation: IEEE 754 floating-point arithmetic is non-associative, meaning that mathematically equivalent computations can produce different results depending on the order of operations, which varies with GPU thread scheduling and batch composition. Full routing reproducibility (MAI-C2) requires batch-invariant computation at significant throughput cost. The three-tier conformance structure reflects this reality.

2 The MoE Stochasticity Problem

2.1 Dense Models: Static Computational Graphs

In a standard dense transformer, every token follows an identical computational path through every layer. The audit surface for a dense model is fully characterized by:

$$\text{Audit Surface}_{\text{dense}} = \{W_1, W_2, \ldots, W_L\} \tag{1}$$

where $\{W_\ell\}$ are the weight matrices at each layer ℓ. Given the weights and the input, the output is a deterministic function (subject only to floating-point precision effects). The dense model's governance properties follow from weight integrity alone: if the weights are certified, every token's computational path is implicitly certified.

2.2 MoE Models: Dynamic Routing as Audit Gap

In an MoE transformer, each MoE layer replaces the single feed-forward network (FFN) with N expert FFNs and a learned gating function G that selects K experts per token. The audit surface expands to:

$$\text{Audit Surface}_{\text{MoE}} = \{W_\ell\} \cup \{G_\ell, R_\ell(h_t) \; \forall \; t, \ell\} \tag{2}$$

where G_ℓ is the gating function at layer ℓ and $R_\ell(h_t)$ is the routing decision (the set of selected experts and their gating weights) for token t at layer ℓ. Weight certification is *necessary but insufficient*: two MoE systems with identical weights can produce different outputs for the same input if their routing decisions differ.

2.3 The Combinatorial Scale of Routing Non-Determinism

The routing decision space grows combinatorially with model scale. For a model with N experts per layer, K experts selected per token, L MoE layers, and a sequence of T tokens, the total number of possible routing configurations is:

$$|\mathcal{R}| = \left[\binom{N}{K}\right]^{T \times L} \tag{3}$$

For DeepSeek-V3 ($N = 256$, $K = 8$, $L = 58$, $T = 4096$):

$$\binom{256}{8} \approx 4.4 \times 10^{13} \tag{4}$$

yielding $|\mathcal{R}| = (4.4 \times 10^{13})^{237,568} \approx 10^{3.2 \times 10^6}$ possible routing configurations per request. Even in the simplified case where routing decisions are independent across layers and tokens (which they are not, due to the sequential dependence of hidden states), the space is astronomically large.

2.4 Three Sources of Routing Non-Determinism

Analysis of the four reference architectures reveals three distinct sources of routing non-determinism, each with different severity and mitigation cost:

2.4.1 Source 1: Algorithmic Stochasticity (Training-Only)

Several MoE architectures introduce explicit randomness into the routing function during training to encourage expert exploration and prevent routing collapse:

- **Noisy top-k gating** (Shazeer et al., 2017): Adds Gaussian noise to router logits before the top-k selection.
- **Input jitter** (Switch Transformer): Multiplies the token representation by a uniform random scalar $\in [1 - \epsilon, 1 + \epsilon]$ before the router projection.
- **Random second-expert selection** (GShard): Selects the second expert randomly with probability proportional to gating weight.

Status: SOLVED. All four reference architectures disable stochastic routing mechanisms at inference time. Inference routing is deterministic as a function of the router weights and the token representation. This is confirmed in the technical reports for DeepSeek-V3, Mixtral, Switch Transformer, and ST-MoE. The attestation requirement (NR-DET-01) codifies this existing practice.

2.4.2 Source 2: Capacity-Induced Token Dropping (Architecture-Dependent)

Switch Transformer-class architectures impose a capacity buffer on each expert:

$$\text{ExpertCapacity} = \left\lfloor \frac{T}{N} \times \text{CF} \right\rfloor \tag{5}$$

where CF is the capacity factor (typically 1.0–1.5). Tokens assigned to an expert that has reached capacity are *dropped*: they bypass the expert layer entirely, passing through only the residual connection. This creates a "Routing Execution Gap"—the router's intended assignment is not honored.

Token dropping is *batch-dependent*: whether a token is dropped depends not only on its own routing decision but on the routing decisions of all other tokens in the batch. The same token may be processed by its intended expert in one batch composition and dropped in another.

Status: SOLVABLE. Architectures without capacity factors (Mixtral, DeepSeek-V3) do not exhibit this problem. For capacity-based architectures, dropless MoE implementations (MegaBlocks; Gale et al., MLSys 2023) eliminate token dropping through block-sparse reformulation. The attestation requirement (NR-DROP-01) mandates zero token dropping during attested inference.

2.4.3 Source 3: Floating-Point Non-Determinism (Fundamental)

IEEE 754 floating-point arithmetic is non-associative:

$$(a + b) + c \neq a + (b + c) \quad \text{in general} \tag{6}$$

GPU matrix multiplication and reduction operations sum partial products in an order determined by thread scheduling, which varies between runs. This means:

- The token representation h_t entering the router may differ across runs of the same input (due to non-deterministic attention and normalization computations in preceding layers).
- The router logits $s = W_r \cdot h_t$ may differ even for identical h_t (due to non-deterministic matrix-vector multiplication).
- These differences are typically small (on the order of machine epsilon) but can change the top-k selection when expert scores are close—which is precisely the regime where load balancing mechanisms push them.

Empirical evidence: Experiments on Qwen3-235B-A22B (an MoE model with 128 experts, top-8 routing) demonstrated that 1,000 identical prompts processed at temperature 0 produced 80 unique completions due to floating-point non-determinism in the routing path.

Status: HARD. Full mitigation requires batch-invariant GPU kernels that enforce deterministic accumulation order, at an estimated throughput cost of 30–40%. The three-tier conformance structure (C0/C1/C2) reflects this cost: C0 and C1 accept floating-point non-determinism; C2 requires batch-invariant computation.

2.5 The Attestation Tax: Cost of Routing Determinism

Table 1 summarizes the mitigation cost for each source of routing non-determinism.

Table 1: Attestation tax: cost of mitigating each non-determinism source.

Mitigation	Throughput Cost	Conformance Level	Notes
Disable training-time noise	Zero	C0	Already standard practice at inference.
FP32 router casting	<5%	C1 (recommended)	Router is <0.01% of total FLOPs.
Canonical tie-breaking	Negligible	C0	Single comparison per top-k call.
Dropless routing	Architecture-dependent	C0	Free for Mixtral/DeepSeek-V3; requires MegaBlocks for Switch-class.
Batch-invariant kernels	∼30–40%	C2	Deterministic accumulation order across all layers.

3 Routing Architecture Analysis

This section analyzes the routing mechanisms of four representative MoE architectures, extracting the properties relevant to attestation design. Each subsection identifies the gating function, selection mechanism, load balancing strategy, and the specific attestation challenges and opportunities each architecture presents.

3.1 DeepSeek-V3: Sigmoid Gating with Auxiliary-Loss-Free Balancing

DeepSeek-V3 (arXiv:2412.19437, December 2024) employs 256 routed experts plus 1 shared (always-active) expert per MoE layer, with top-8 selection across 58 MoE layers (of 61 total transformer layers). Total parameters: 671B; active parameters per token: approximately 37B.

3.1.1 Sigmoid Independence Property

DeepSeek-V3's most attestation-relevant innovation is its use of *sigmoid* gating rather than softmax. The affinity score for each expert is computed independently:

$$s_i = \sigma(W_r^{(i)} \cdot h_t + b_i) \tag{7}$$

where σ is the sigmoid function, $W_r^{(i)}$ is the router weight vector for expert i, h_t is the token representation, and b_i is the load-balancing bias term. Because each s_i is computed independently (no shared denominator as in softmax), a verifier can check individual expert scores without access to the full score vector.

Attestation implication. Sigmoid independence enables *sharded verification*: a verifier with access to only one expert's router weights and bias can verify that expert's selection decision for a given token. This is not possible with softmax gating, where all expert scores are coupled through the normalization denominator.

3.1.2 Auxiliary-Loss-Free Bias Balancing

DeepSeek-V3 replaces the traditional auxiliary load-balancing loss with a deterministic bias update mechanism. Each expert i maintains a bias term b_i that is adjusted during training by a simple control loop:

> If expert i is overloaded (receives more than its fair share of tokens in the current batch), decrease b_i by γ. If underloaded, increase b_i by γ.

where $\gamma = 0.001$ is the bias update speed. At inference time, the bias vector $\mathbf{b} \in \mathbb{R}^{256}$ is *frozen* and serves as a static component of the routing function.

Attestation implication. The frozen bias vector is an interpretable, low-dimensional attestation artifact. Its values directly encode the load-balancing state learned during training: experts with large negative bias were persistently overloaded (the bias was decreased to discourage further routing to them), while experts with large positive bias were underutilized. The variance of the bias vector provides a scalar summary of load imbalance severity. This is a natural "Load Balancing State Manifest" that can be disclosed, hashed, and bound to the model identity.

3.1.3 Node-Limited Routing

DeepSeek-V3 deploys its 256 experts across 8 physical nodes (32 experts per node). To limit cross-node communication, the routing mechanism constrains each token to select experts from at most $M = 4$ of the 8 nodes. The node selection is performed by aggregating per-expert scores at the node level and selecting the top-M nodes before performing expert selection within the selected nodes.

Attestation implication. Node-limited routing introduces a hardware-induced routing bias: a token's expert selection is constrained by physical topology, not solely by learned affinity scores. The globally highest-scoring expert may be excluded if its node was not among the top-M. This means the attestation system must disclose the expert-to-node mapping and log the node selection vector, enabling a verifier to distinguish between routing choices driven by learned affinity and routing choices driven by topology constraints.

3.1.4 Shared Expert Baseline

The shared expert processes every token regardless of routing decisions. Its output is combined with the routed experts' outputs:

$$y_t = y_t^{\text{shared}} + \sum_{k=1}^{K} g_{k,t} \cdot y_{k,t}^{\text{routed}} \tag{8}$$

Attestation implication. The shared expert provides a routing-independent baseline. Even if routing decisions change (due to floating-point non-determinism or other factors), the shared expert's contribution to the output is constant. This creates a natural decomposition: the fraction of output magnitude attributable to the shared expert versus the routed experts quantifies the *routing sensitivity* of the model's output for each token.

3.2 Mixtral 8x7B: Softmax Gating with Top-2 Selection

Mixtral 8x7B (arXiv:2401.04088, January 2024) uses 8 experts per MoE layer with top-2 selection across 32 MoE layers. Total parameters: approximately 47B; active parameters per token: approximately 13B.

3.2.1 Softmax Coupling

Mixtral uses standard softmax gating:

$$p_i = \frac{\exp(W_r^{(i)} \cdot h_t)}{\sum_{j=1}^{8} \exp(W_r^{(j)} \cdot h_t)} \tag{9}$$

The top-2 experts are selected by the highest p_i values, and the gating weights for the selected experts are renormalized to sum to 1.

Attestation implication. Unlike DeepSeek-V3's sigmoid gating, softmax couples all expert scores through the shared denominator $\sum_j \exp(\cdot)$. A verifier cannot check a single expert's score independently—all 8 logits must be available to verify any individual gating weight. This means that at C2-level verification, the full logit vector must be logged for every routing decision.

3.2.2 Implementation Discrepancy

A notable discrepancy exists between Mixtral's paper description and its reference implementation. The HuggingFace implementation applies softmax *before* top-k selection, then renormalizes the selected experts' weights—rather than selecting top-k from raw logits and then applying softmax only to the selected experts. The two procedures can produce different results when the softmax distribution is flat (all experts have similar scores), because the renormalization step changes the relative weights.

Attestation implication. The gating computation order must be explicitly declared in the attestation metadata. A verifier checking routing logs must know whether the system implements "topk-then-softmax" (paper) or "softmax-then-topk-renorm" (HuggingFace) to correctly verify gating weight consistency.

3.2.3 Syntactic Expert Specialization

Analysis of Mixtral's routing patterns reveals that expert specialization is *syntactic rather than semantic*: experts specialize in token types (punctuation, operators, sentence-initial words) rather than knowledge domains (science, law, code). Consecutive tokens share the same first-choice expert 24–28% of the time at layer 15, compared to the 12.5% expected under random assignment.

Attestation implication. Semantic expert labeling (e.g., designating an expert as the "safety expert" or "factual expert") is not supported by the empirical evidence. Governance frameworks that assume domain-specific expert specialization are building on an incorrect foundation. Routing attestation should treat expert selection as a learned computational optimization, not as a semantic routing decision.

3.3 Switch Transformer: Top-1 Routing with Capacity Factor

The Switch Transformer (Fedus et al., JMLR 2022; arXiv:2101.03961) routes each token to a single expert (top-1) with a capacity factor mechanism that limits the number of tokens each expert can process per batch.

3.3.1 Capacity Factor and Token Dropping

Each expert has a buffer of fixed size:

$$\text{ExpertCapacity} = \left\lfloor \frac{T_{\text{batch}}}{N} \times \text{CF} \right\rfloor \tag{10}$$

where T_{batch} is the total number of tokens in the batch, N is the number of experts, and CF is the capacity factor. When more tokens are routed to an expert than its capacity allows, the excess tokens are *dropped*: they bypass the expert layer entirely, passing through only the residual connection.

Attestation implication. Token dropping creates a "Routing Execution Gap": the router intended the token to be processed by Expert i, but the token was not processed by any expert. The output for a dropped token depends only on the residual connection, meaning the MoE layer had no effect on that token. This is invisible in the model's output unless explicitly logged. Furthermore, token dropping is *batch-dependent*: whether a token is dropped depends on all other routing decisions in the batch. The same prompt processed in two different batch compositions may produce different outputs solely because different tokens were dropped.

3.3.2 Auxiliary Load-Balancing Loss

Switch Transformer uses an auxiliary loss to encourage balanced expert utilization:

$$\mathcal{L}_{\text{balance}} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{11}$$

where f_i is the fraction of tokens routed to expert i, P_i is the average router probability assigned to expert i, and $\alpha = 10^{-2}$ is the balancing coefficient. This loss operates only during training; at inference, its effect is embedded in the learned router weights.

3.3.3 FP32 Router Recommendation

The Switch Transformer paper explicitly recommends performing the router computation in float32 precision even when the rest of the model operates in bfloat16: "We also cast the router input to float32 to ensure stability." This recommendation is motivated by the observation that low-precision router computation can cause routing oscillations where tokens flip between experts across consecutive forward passes.

Attestation implication. Router precision is an attestation-relevant property. The normative requirements (Section 6) include router precision disclosure at all conformance levels and recommend FP32 router computation at MAI-C1 and above.

3.4 ST-MoE: Z-Loss for Numerical Stability

ST-MoE (Zoph et al., arXiv:2202.08906, 2022) introduces the router z-loss, a regularization term that penalizes large router logits:

$$\mathcal{L}_z = \frac{1}{B} \sum_{x=1}^{B} \left(\log \sum_{j=1}^{N} \exp(x_j) \right)^2 \tag{12}$$

3.4.1 BF16 Roundoff Analysis

ST-MoE provides the most detailed analysis of precision effects on routing stability. BF16 (brain floating-point 16) has only 7 bits of mantissa, compared to FP16's 10 bits and FP32's 23 bits. For large logit values, BF16 quantization can cause catastrophic changes in the softmax distribution.

The ST-MoE analysis demonstrates a concrete example: a roundoff error of 0.5 on a single logit (well within BF16's quantization range for logit magnitudes > 64) can change the softmax output by up to 36%. The z-loss prevents this by keeping logit magnitudes small, ensuring that the BF16 quantization grid is fine enough to preserve the relative ordering of expert scores.

Attestation implication. Systems operating routers in BF16 without z-loss (or equivalent logit magnitude control) are at elevated risk of precision-induced routing instability. The normative requirements include a logit magnitude monitoring provision: systems at MAI-C1 and above that operate routers below FP32 precision must disclose the maximum observed logit magnitude and whether z-loss was used during training.

3.5 Cross-Architecture Attestation Summary

Table 2 summarizes the attestation-relevant properties across the four reference architectures.

Table 2: Cross-architecture attestation property summary.

Property	**DeepSeek-V3**	**Mixtral**	**Switch**	**ST-MoE**
Inference deterministic	Yes	Yes	Yes	Yes
Independent score verification	Yes (sigmoid)	No (softmax)	No (softmax)	No (softmax)
Token dropping risk	None	None	Yes (CF-based)	Yes (CF-based)
Load balancing artifact	Bias vector	None	Aux loss (embedded in weights)	Aux + z-loss (embedded)
Topology constraints	Yes (node-limited)	No	No	No
Shared expert baseline	Yes	No	No	No
Router precision recommendation	Not specified	Not specified	FP32 explicit	Z-loss (implicit)

4 The Determinism Landscape

4.1 Inference-Time Routing Is Already Deterministic

A central empirical finding of this specification is that **all major MoE architectures already use deterministic routing at inference time**. The stochastic mechanisms documented in the literature—noisy top-k gating, input jitter, random second-expert selection, Gumbel noise—are *training-time only* and are disabled during inference.

Table 3 surveys the routing determinism status across the major MoE architectures published between 2017 and 2024.

Table 3: Inference-time routing determinism across MoE architectures.

Architecture	**Training Noise**	**Inference Routing**	**Source**
Shazeer et al. (2017)	Gaussian noisy top-k	Deterministic (noise = 0)	ICLR 2017
GShard (2020)	Random 2nd expert	Deterministic (top-2 argmax)	arXiv:2006.16668
Switch (2022)	Input jitter $U[1-\epsilon, 1+\epsilon]$	Deterministic (jitter disabled)	JMLR 2022
ST-MoE (2022)	Jitter + z-loss	Deterministic	arXiv:2202.08906
Hash Routing (2021)	None	Deterministic (hash function)	NeurIPS 2021
BASE Layers (2021)	None	Deterministic (linear assignment)	ICML 2021
Expert Choice (2022)	None	Deterministic (expert selects top-k tokens)	NeurIPS 2022
Mixtral (2024)	None	Deterministic (softmax top-2)	arXiv:2401.04088
DeepSeek-V3 (2024)	Bias only (deterministic)	Deterministic (sigmoid top-8)	arXiv:2412.19437

Implication for attestation design. The attestation problem for MoE routing is *not* algorithmic stochasticity. The algorithms are already deterministic. The attestation problem is (1) documenting and verifying that determinism, (2) addressing floating-point non-determinism in the underlying GPU computation, and (3) logging routing decisions for forensic reconstruction.

4.2 Floating-Point Non-Determinism: The Residual Problem

With algorithmic stochasticity eliminated, the remaining source of routing non-determinism is the non-associativity of IEEE 754 floating-point arithmetic.

4.2.1 Non-Associative Reduction Operations

GPU matrix multiplication and reduction operations (softmax denominators, layer normalization statistics) compute sums over large vectors. The order in which partial sums are accumulated depends on thread scheduling, which is not deterministic across runs. Because floating-point addition is non-associative:

$$\text{fl}((a+b)+c) \neq \text{fl}(a+(b+c)) \quad \text{in general} \tag{13}$$

the result of a reduction over n values can vary by up to $O(n \cdot \epsilon_{\text{mach}})$ across runs, where ϵ_{mach} is the machine epsilon for the operational precision ($\epsilon_{\text{mach}} \approx 3.9 \times 10^{-3}$ for BF16, 9.8×10^{-4} for FP16, 1.2×10^{-7} for FP32).

4.2.2 Batch-Composition Dependence

In batched inference, multiple requests share computation through batched matrix multiplications and attention operations. The hidden state h_t for a token in request A is computed by operations that also process tokens from requests B, C, etc. Because the batched computation is non-deterministic (due to the reduction order effects above), the hidden state h_t—and therefore the router input—depends on which other requests are in the batch.

This means that the same prompt, submitted twice to the same model on the same hardware, can produce different routing decisions if the co-occurring requests in the batch differ.

4.2.3 Precision Hierarchy

The severity of floating-point non-determinism depends on the operational precision:

- **FP32** (23-bit mantissa): Most stable. Differences between runs are typically below the threshold that would change a top-k selection, except in pathological cases where expert scores are near-identical.
- **FP16** (10-bit mantissa): Moderate stability. The coarser quantization grid increases the frequency of ties and near-ties.
- **BF16** (7-bit mantissa): Least stable. The very coarse quantization grid makes ties frequent, and the ST-MoE analysis shows that roundoff errors can change softmax distributions by up to 36%.

4.3 MAI Conformance Levels for Routing Determinism

Based on the analysis above, the three MAI conformance levels correspond to progressively stronger guarantees about routing determinism:

Table 4: MAI conformance levels for routing determinism.

Level	Routing Determinism Guarantee
MAI-C0	Algorithmic determinism confirmed. Training-time noise disabled. Canonical tie-breaking implemented. Router precision disclosed. No guarantee against floating-point non-determinism.
MAI-C1	All C0 guarantees, plus: FP32 router computation recommended. Routing decisions logged at $\geq$1% sampling rate. Expert utilization monitored with anomaly thresholds. Deployment topology disclosed.
MAI-C2	All C1 guarantees, plus: batch-invariant routing required. 100% routing decisions logged. Node selection logged. Bias vector cryptographically bound. Per-request routing reproducibility demonstrated.

5 Expert Parallelism and Distributed Attestation

Production MoE deployments distribute experts across multiple GPUs and physical nodes. This distributed execution creates a fundamental attestation challenge: no single GPU possesses the complete routing and computation state.

5.1 The All-to-All Communication Protocol

Expert parallelism follows a four-phase protocol:

1. **Local routing (Routing GPU):** Each GPU computes the routing function for its local token shard, determining which experts each token should be sent to.
2. **Token dispatch (All-to-All):** Tokens are sent from their source GPU to the GPU hosting their assigned expert(s). This is an all-to-all communication pattern: every GPU may send tokens to every other GPU.
3. **Expert computation (Expert GPU):** Each GPU executes its local experts on the received tokens.
4. **Result gather (All-to-All):** Expert outputs are sent back to the source GPUs, where they are combined using the gating weights.

5.2 The Cross-GPU Verification Gap

The four-phase protocol creates an attestation asymmetry between the routing GPU and the expert GPU:

- The **routing GPU** knows *why* a token was sent to a particular expert (it has the full affinity scores, the gating weights, and the selection logic). But it does not observe the expert's computation.
- The **expert GPU** knows *what* computation was performed on the token (it has the expert weights and the token representation). But it does not know *why* the token was sent to it—it does not have the routing scores.

Neither GPU can independently verify the complete routing-to-computation pipeline. This is the cross-GPU verification gap.

5.3 Communication Patterns by Framework

The major MoE serving frameworks implement the all-to-all protocol with different optimization strategies:

- **Megatron-LM:** Standard All-to-All dispatch with communication-computation overlap (expert computation begins as tokens arrive, before the full dispatch is complete).
- **DeepSpeed-MoE:** Hierarchical All-to-All that separates intra-node (NVLink) and inter-node (InfiniBand) communication, achieving up to 7.3× better latency than flat All-to-All on 128-GPU clusters (Rajbhandari et al., ICML 2022).
- **FairSeq:** Basic All-to-All dispatch without hierarchical optimization.

Communication latency. Cross-node all-to-all communication incurs 1–5 ms latency (InfiniBand), compared to 10–50 μs for intra-node NVLink transfers. For DeepSeek-V3 with 8 nodes, the all-to-all dispatch dominates MoE layer latency.

5.4 Potential Mitigations for Cross-GPU Verification

Expert weight hash verification. Each GPU hosting experts can compute and attest the SHA-256 hash of its expert weights, demonstrating that the correct expert parameters were loaded. This does not verify that the correct expert *processed* a given token, but it does verify that the expert was not corrupted or substituted.

TEE-attested expert execution. If both the routing GPU and expert GPU operate within Trusted Execution Environments (TEEs), the TEE attestation chain provides hardware-level assurance that the declared code was executed on the declared hardware. The cross-GPU verification gap is narrowed (though not closed) because each GPU's TEE attests its local computation. This requires TEE support on all GPUs in the deployment—a capability that is available on NVIDIA H100 (Confidential Computing mode) but not universally deployed.

Spot-check verification. For a randomly sampled subset of routing decisions, the serving system replays the expert computation on the routing GPU (which has the token representation) using a copy of the expert weights. If the replayed output matches the expert GPU's output, confidence in the routing-to-computation binding increases. This approach is probabilistic rather than deterministic and incurs additional compute cost proportional to the sampling rate.

Output consistency checks. Statistical consistency checks can detect gross routing-computation mismatches: if Expert i consistently produces outputs that are statistically dissimilar to its validation-time output distribution, this may indicate that Expert i was substituted, corrupted, or that tokens intended for Expert i were redirected to a different expert.

5.5 Communication Integrity

The all-to-all dispatch relies on the GPU communication library (NCCL for NVIDIA GPUs) to deliver tokens to the correct destination. NCCL provides reliable, ordered delivery but does not provide cryptographic integrity verification. A compromised network switch or a bit-flip in transit could alter token representations without detection.

Attestation recommendation. At MAI-C2, the communication integrity between GPUs involved in expert parallelism **SHOULD** be verified through integrity checks on the all-to-all dispatch. The recommended approach is CRC-32 on each dispatch message for performance-sensitive deployments, or SHA-256 for high-assurance deployments. This adds modest overhead (CRC-32: <1% of communication latency) but closes the integrity gap in the all-to-all protocol.

Honest Framing

The cross-GPU verification gap is an honest limitation of current MoE attestation infrastructure. No existing serving framework (Megatron-LM, DeepSpeed-MoE, FairSeq) provides built-in cryptographic proof that a specific expert executed on a specific token. The mitigations described above reduce the gap but do not close it entirely. Full closure would require either (a) TEE attestation on all GPUs with intra-TEE token tracking, or (b) cryptographic protocols that bind routing decisions to expert computations at the token level—both of which are open research problems.
This specification acknowledges the gap and provides the requirements for the closest achievable approximation on current hardware: weight hash verification, topology disclosure, and sampled spot-check protocols.

6 Normative Requirements

This section defines the mandatory and recommended requirements for MoE routing attestation within the Auburn Governance Stack. Requirements are organized by functional category and tagged with their applicable MAI-1 conformance level(s). All requirements use RFC 2119/8174 normative language.

Requirements tagged **[C0]** apply at MAI-C0 (Minimum) and all higher levels. Requirements tagged **[C1]** apply at MAI-C1 (Standard) and MAI-C2 (Full). Requirements tagged **[C2]** apply only at MAI-C2 (Full).

6.1 Deterministic Routing Mode

6.1.1 Noise Removal at Inference

NR-DET-01: Training-Time Noise Disabled [C0]

Systems claiming MAI-1 conformance at any level **SHALL** disable all training-time stochastic routing mechanisms during attested inference. This includes, but is not limited to:

- Gaussian noise injection on router logits (Shazeer et al., 2017).
- Multiplicative input jitter on token representations (Fedus et al., 2022).
- Random expert selection for secondary routing slots (GShard).
- Gumbel noise for differentiable top-k approximation.
- Any other mechanism that introduces non-deterministic variation into the routing selection process.

The attestation metadata **SHALL** include a Boolean field `routing-stochasticity-disabled` set to `true`. Systems that retain any stochastic routing mechanism during inference **MUST NOT** claim MAI-1 conformance at any level.

NR-DET-02: Deterministic Selection Function [C0]

The expert selection function **SHALL** be a deterministic function of the router weights, the token representation, and (where applicable) the load-balancing bias vector. For a given input tuple $(W_r, h_t, \mathbf{b})$, the selected expert set and gating weights **SHALL** be identical across invocations, subject only to floating-point precision effects addressed in NR-DET-04 through NR-DET-06.
Acceptable deterministic selection functions include:

- Argmax (top-1 selection): $i^* = \text{argmax}_i \; f(W_r, h_t, \mathbf{b})_i$
- Top-K selection: $\{i_1, \ldots, i_K\} = \text{TopK}_i \; f(W_r, h_t, \mathbf{b})_i$
- Hash-based routing: $i^* = \text{hash}(\text{token\_id}) \mod N$
- Linear assignment (BASE Layers): deterministic optimization-based assignment

6.1.2 Canonical Tie-Breaking Rule

NR-DET-03: Canonical Tie-Breaking [C0]

Systems claiming MAI-1 conformance **SHALL** implement and document a deterministic tie-breaking rule for the top-k expert selection. The tie-breaking rule **SHALL** produce identical results regardless of thread scheduling, memory layout, or GPU architecture. The *recommended* canonical tie-breaking rule is:

> *In the event that two or more experts receive identical affinity scores after rounding to the router's operational precision, select the expert with the lowest numerical index.*

Systems that implement an alternative tie-breaking rule (e.g., highest index, round-robin, hash-based) **MAY** do so provided the rule is:

(a) Deterministic (produces identical results for identical inputs).

(b) Documented in the attestation metadata.

(c) Independent of runtime state (thread scheduling, batch composition, memory layout).

The implemented tie-breaking rule **SHALL** be recorded in the attestation metadata field `routing-tiebreak-rule`.

Rationale. The tie-breaking problem is not theoretical. In BF16 precision (7 bits of mantissa), the quantization grid is coarse: values differing by less than $2^{-7} \approx 0.0078$ relative to their magnitude are rounded to the same representation. For a router operating on 256 experts (DeepSeek-V3), the probability that at least two experts receive the same BF16-rounded score for a given token is non-negligible, particularly in early layers where router weights may not yet have diverged sufficiently. Standard GPU top-k implementations (e.g., PyTorch's `torch.topk`) resolve ties through implementation-defined behavior that may depend on input memory layout, which varies with batch size and padding. The canonical tie-breaking rule eliminates this source of non-reproducibility.

6.1.3 Router Precision Requirements

NR-DET-04: Router Precision Disclosure [C0]

Systems claiming MAI-1 conformance **SHALL** disclose the numerical precision used for each stage of the routing computation in the attestation metadata:

- `router-projection-precision`: Precision of the linear projection $s = W_r \cdot h_t$ (e.g., `float32`, `float16`, `bfloat16`).
- `router-gating-precision`: Precision of the gating function (softmax, sigmoid, or other normalization).
- `router-selection-precision`: Precision of the top-k selection and tie-breaking logic.
- `expert-computation-precision`: Precision of the expert feed-forward computation (reported separately, as it may differ from router precision).

NR-DET-05: FP32 Router Computation [C1]

Systems claiming MAI-1 conformance at MAI-C1 or above **SHOULD** perform the router linear projection and gating function computation in `float32` precision, regardless of the precision used for expert computation.
This recommendation follows the Switch Transformer's explicit guidance (Section 2.4: "We also cast the router input to float32 to ensure stability") and is motivated by the ST-MoE analysis demonstrating that BF16 roundoff errors can alter softmax routing distributions by up to 36% when logits are large (§3.4.1).
Systems that operate the router in a precision lower than `float32` **SHALL** additionally disclose:

- The maximum observed router logit magnitude during model validation.
- A statement of whether the router was trained with z-loss regularization (or equivalent logit magnitude control).

NR-DET-06: Batch-Invariant Routing [C2]

Systems claiming MAI-1 conformance at MAI-C2 **SHALL** ensure that routing decisions for a given token are invariant to the composition of the serving batch. Formally: for a token t with representation h_t computed from prompt P, the routing decision $R(h_t)$ **SHALL** be identical whether P is processed in isolation, in a batch with prompts $\{P_1, \ldots, P_m\}$, or in a batch with prompts $\{Q_1, \ldots, Q_n\}$.
This requirement may be satisfied by any of the following mechanisms:

(a) **Batch-invariant GPU kernels:** Matrix multiplication, normalization, and reduction operations that produce bitwise-identical results regardless of batch composition.

(b) **Single-request batching:** Processing each request in isolation (batch size 1), eliminating batch-composition effects at the cost of throughput.

(c) **Deterministic batch construction:** Using a fixed padding scheme that makes the effective batch composition constant regardless of actual request mix.

(d) **FP32 full-precision inference:** Operating the entire model (not only the router) in `float32`, which empirically reduces floating-point non-determinism to near-zero levels.

The mechanism used to achieve batch-invariant routing **SHALL** be documented in the attestation metadata field `batch-invariance-mechanism`. Systems at MAI-C2 **SHALL** demonstrate batch-invariance through a reproducibility test: the same prompt processed in 100 different batch compositions **SHALL** produce identical routing decisions across all 100 runs.

6.2 Dropless Execution

6.2.1 Zero Token Dropping Requirement

NR-DROP-01: Dropless Inference [C0]

Systems claiming MAI-1 conformance at any level **SHALL** guarantee that no tokens are dropped due to expert capacity overflow during attested inference. The token drop rate **SHALL** be exactly 0%.
This requirement is satisfied by:

(a) Architectures that do not use capacity factors (e.g., Mixtral, DeepSeek-V3), where expert buffers are dynamically sized or effectively unbounded.

(b) Architectures that use capacity factors but set the capacity factor sufficiently high to guarantee zero drops for the maximum expected batch size.

(c) Block-sparse MoE implementations (e.g., MegaBlocks; Gale et al., MLSys 2023) that reformulate expert computation to eliminate the token-dropping mechanism entirely.

Systems that cannot guarantee zero token dropping **SHALL** disclose the capacity factor, the maximum expected batch size, and the empirically observed maximum token drop rate during validation. A token drop rate $> 0\%$ during attested inference constitutes a *routing integrity violation* at all conformance levels.

6.2.2 Capacity Factor Disclosure

NR-DROP-02: Capacity Factor Disclosure [C0]

Systems that employ a capacity factor mechanism (Switch Transformer-class architectures) **SHALL** disclose the following in the attestation metadata:

- The capacity factor value (CF).
- The resulting expert buffer size for the declared maximum batch size.
- The behavior when expert capacity is exceeded: token dropping (residual-only bypass), token rerouting (to next-best expert), or other.
- The empirically observed maximum token drop rate across the validation dataset.

Systems that do not employ a capacity factor mechanism (e.g., Mixtral, DeepSeek-V3) **SHALL** declare `capacity-factor: null` and `token-dropping: false` in the attestation metadata.

6.3 Deployment Topology Disclosure

6.3.1 Expert-to-GPU Mapping

NR-TOPO-01: Expert Assignment Table [C1]

Systems claiming MAI-1 conformance at MAI-C1 or above **SHALL** include in the attestation evidence an expert assignment table mapping each expert index to its physical GPU identifier. The table **SHALL** include:

- `expert-id`: The logical expert index $(0, \ldots, N-1)$.
- `gpu-id`: The physical GPU identifier (device UUID or platform-attested device identity per MAI-1 Layer 1).
- `node-id`: The physical node identifier (for multi-node deployments).
- `replica-flag`: Boolean indicating whether this is a primary or replica instance of the expert (for redundant expert deployments).

The expert assignment table **SHALL** be cryptographically bound to the Layer 1 platform attestation evidence through inclusion in the MAI-1 Merkle tree (per Document 24, Cryptographic Binding Specification).

NR-TOPO-02: Topology Change Notification [C1]

Any change to the expert-to-GPU mapping (due to GPU failure, load rebalancing, expert migration, or cluster reconfiguration) **SHALL** trigger a fresh attestation cycle. The previous attestation evidence **SHALL** be invalidated, and new evidence reflecting the updated topology **SHALL** be generated before attested inference resumes.
The attestation metadata **SHALL** include a monotonically increasing `topology-version` counter that increments on every topology change.

6.3.2 Node-Limited Routing Constraints

NR-TOPO-03: Node Constraint Disclosure [C1]

Systems employing node-limited or topology-constrained routing (e.g., DeepSeek-V3's M-of-N node selection) **SHALL** disclose:

- The node grouping: which experts reside on which physical node.
- The maximum number of nodes selectable per token (M).
- The node selection algorithm (e.g., aggregate top-M by summed per-group affinity scores).
- The total number of nodes (N_{nodes}) in the deployment.

NR-TOPO-04: Node Selection Logging [C2]

Systems at MAI-C2 that employ node-limited routing **SHALL** log the node selection vector for each token in the Routing Attestation Log. The node selection vector records which nodes were selected and the aggregate score for each node, enabling verification that topology constraints—rather than routing failures—explain any deviation between the globally highest-affinity experts and the actually-selected experts.

6.4 Load Balancing State Attestation

6.4.1 Bias Vector Disclosure (DeepSeek-Style Architectures)

NR-LOAD-01: Bias Vector Disclosure [C1]

Systems employing bias-based load balancing (DeepSeek-V3 ALF-style or equivalent) **SHALL** disclose the frozen bias vector $\mathbf{b} \in \mathbb{R}^N$ used during inference as part of the attestation metadata. The bias vector **SHALL** be disclosed in full (all N values) and **SHALL** be cryptographically bound to the model identity hash.
The attestation metadata **SHALL** additionally include:

- `bias-update-speed`: The γ hyperparameter used during training (for interpretability of bias magnitudes).
- `bias-vector-variance`: $\mathrm{Var}(\mathbf{b})$, as a scalar summary of load imbalance severity.
- `bias-vector-range`: $[\min(\mathbf{b}), \max(\mathbf{b})]$.

NR-LOAD-02: Bias Vector Integrity [C2]

At MAI-C2, the bias vector **SHALL** be included in the model's Merkle tree hash (alongside the router weights and expert weights). Any modification to the bias vector **SHALL** invalidate the current attestation evidence and require a fresh attestation cycle.
The bias vector **MUST NOT** be modified during inference. Systems that implement dynamic bias adjustment at inference time (e.g., for real-time load rebalancing) **MUST NOT** claim MAI-C2 conformance. Dynamic inference-time bias adjustment is permitted at MAI-C0 and MAI-C1 provided the adjustment mechanism and current bias state are logged.

6.4.2 Expert Utilization Distribution

NR-LOAD-03: Utilization Distribution Reporting [C0]

Systems claiming MAI-1 conformance at any level **SHALL** report the expert utilization distribution in the attestation payload. The utilization distribution **SHALL** include:

- `utilization-counts`: The number of tokens processed by each expert during the attestation measurement window.
- `utilization-fraction`: The fraction of total tokens processed by each expert ($f_i = \mathrm{count}_i / \sum_j \mathrm{count}_j$).
- `utilization-entropy`: The Shannon entropy of the utilization distribution $H = -\sum_i f_i \log_2 f_i$, as a scalar summary of balance. For perfectly uniform utilization over N experts, $H = \log_2 N$.
- `measurement-window`: The number of tokens and time interval over which the utilization was measured.

NR-LOAD-04: Utilization Anomaly Thresholds [C1]

Systems at MAI-C1 or above **SHALL** define and disclose expert utilization anomaly thresholds. At minimum, the following conditions **SHALL** be monitored and reported:

- **Dead expert:** An expert with utilization fraction $f_i < \epsilon_{\text{dead}}$ over a measurement window, where ϵ_{dead} is a configurable threshold (recommended: $\epsilon_{\text{dead}} = \frac{0.01}{N}$, i.e., less than 1% of the uniform expectation). A dead expert indicates potential routing collapse or expert degradation.
- **Dominant expert:** An expert with utilization fraction $f_i > \kappa \cdot \frac{1}{N}$ over a measurement window, where κ is a configurable multiplier (recommended: $\kappa = 10$, i.e., more than $10\times$ the uniform expectation). A dominant expert indicates routing concentration that may impair model diversity.
- **Utilization entropy collapse:** $H < H_{\text{min}}$, where H_{min} is a configurable minimum entropy threshold (recommended: $H_{\text{min}} = 0.7 \cdot \log_2 N$, i.e., at least 70% of maximum entropy). Entropy collapse indicates systematic routing imbalance across the expert population.

The specific threshold values **SHALL** be disclosed in the attestation metadata. Threshold violations **SHALL** be reported as routing health warnings in the MAI-1 attestation payload. Persistent threshold violations (exceeding a configurable duration window) **SHALL** trigger a governance state transition to *review* status per MAI-1 §8 conformance state machine.

6.5 Routing Decision Logging

NR-LOG-01: Routing Metadata Disclosure [C0]

Systems claiming MAI-1 conformance at any level **SHALL** include the following routing metadata in the MAI-1 Layer 2 attestation payload:

- `routing-architecture`: The routing mechanism type (e.g., `sigmoid-topk`, `softmax-topk`, `softmax-top1`, `hash`, `expert-choice`).
- `num-experts`: Total number of routed experts per layer (N).
- `num-shared-experts`: Number of shared (always-active) experts.
- `top-k`: Number of experts selected per token (K).
- `num-moe-layers`: Number of MoE layers in the model.
- `gating-function`: The gating normalization function (`softmax`, `sigmoid-normalize`, `none`).
- `routing-stochasticity-disabled`: Boolean (per NR-DET-01).
- `routing-tiebreak-rule`: String description (per NR-DET-03).
- `token-dropping`: Boolean (per NR-DROP-02).
- `capacity-factor`: Float or `null` (per NR-DROP-02).
- `router-projection-precision`: String (per NR-DET-04).
- `router-gating-precision`: String (per NR-DET-04).

This metadata provides the minimum information necessary for a verifier to understand the routing mechanism without access to the model or its serving infrastructure.

NR-LOG-02: Sampled Routing Decision Logging [C1]

Systems claiming MAI-1 conformance at MAI-C1 **SHALL** log routing decisions at a configurable sampling rate. The minimum sampling rate **SHALL** be 1% of tokens (1 in 100 tokens logged). For each sampled token, the log **SHALL** contain:

- Token sequence index within the request.
- MoE layer index.
- Selected expert indices (K values).
- Gating weights for the selected experts (K values).
- (If applicable) Whether a tie-breaking rule was invoked.
- (If applicable) Whether the token was rerouted due to capacity constraints.

The sampling mechanism **SHALL** be documented and reproducible (e.g., deterministic sampling based on token index modulo a sampling period, not random sampling). The sampling rate **SHALL** be disclosed in the attestation metadata field `routing-log-sampling-rate`.

NR-LOG-03: Full Routing Attestation Log [C2]

Systems claiming MAI-1 conformance at MAI-C2 **SHALL** generate a complete Routing Attestation Log (RAL) for every inference request. The RAL **SHALL** be:

- Generated for 100% of tokens across 100% of MoE layers.
- Cryptographically bound to the inference request through inclusion in the request's attestation Merkle tree.
- Retained for a configurable audit retention period (recommended minimum: 90 days for regulated deployments).
- Accessible to authorized verifiers through a standardized retrieval API.

The RAL **MAY** be stored in compressed form provided that decompression produces the exact original data (lossless compression only). The RAL **MUST NOT** use lossy compression, approximation, or statistical summarization at MAI-C2.

6.6 Conformance Level Summary

Table 5 provides a consolidated view of routing attestation requirements by MAI-1 conformance level.

Table 5: MoE routing attestation requirements by conformance level.

Requirement	C0	C1	C2	Description
NR-DET-01	**MUST**	**MUST**	**MUST**	Noise disabled
NR-DET-02	**MUST**	**MUST**	**MUST**	Deterministic selection function
NR-DET-03	**MUST**	**MUST**	**MUST**	Canonical tie-breaking
NR-DET-04	**MUST**	**MUST**	**MUST**	Router precision disclosure
NR-DET-05	—	**SHOULD**	**SHOULD**	FP32 router computation
NR-DET-06	—	—	**MUST**	Batch-invariant routing
NR-DROP-01	**MUST**	**MUST**	**MUST**	Zero token dropping
NR-DROP-02	**MUST**	**MUST**	**MUST**	Capacity factor disclosure
NR-TOPO-01	—	**MUST**	**MUST**	Expert assignment table
NR-TOPO-02	—	**MUST**	**MUST**	Topology change notification
NR-TOPO-03	—	**MUST**	**MUST**	Node constraint disclosure
NR-TOPO-04	—	—	**MUST**	Node selection logging
NR-LOAD-01	—	**MUST**	**MUST**	Bias vector disclosure
NR-LOAD-02	—	—	**MUST**	Bias vector integrity (Merkle)
NR-LOAD-03	**MUST**	**MUST**	**MUST**	Utilization distribution
NR-LOAD-04	—	**MUST**	**MUST**	Utilization anomaly thresholds
NR-LOG-01	**MUST**	**MUST**	**MUST**	Routing metadata disclosure
NR-LOG-02	—	**MUST**	**MUST**	Sampled routing logging (1%+)
NR-LOG-03	—	—	**MUST**	Full RAL (100% tokens)

Conformance level design rationale. **MAI-C0** establishes the baseline: algorithmic determinism is confirmed, routing mechanism metadata is disclosed, token dropping is prohibited, and expert utilization is monitored. These requirements impose *zero throughput overhead* on existing MoE inference systems that already use deterministic routing (which, per Section 4.1, is all of them). MAI-C0 is achievable today by any production MoE deployment with minimal engineering effort.

MAI-C1 adds deployment transparency: the physical topology is disclosed, expert weights are hash-verified per GPU, load balancing state is attested, and routing decisions are logged at sampled granularity. The primary engineering cost at MAI-C1 is the routing logging infrastructure (NR-LOG-02), which requires instrumentation of the inference pipeline to capture and persist routing decisions. The throughput overhead is modest (estimated $< 5\%$) because only 1% of tokens are logged.

MAI-C2 requires full routing reproducibility: batch-invariant computation, complete routing logs for every token, node selection logging, and cryptographic binding of the bias vector and topology metadata. This is the "gold standard" for routing attestation, intended for high-stakes regulated deployments where the governance value of full reproducibility justifies the throughput cost (estimated 30–40% for batch-invariant computation, plus storage costs for full RAL generation).

Honest Framing

The three-tier conformance structure reflects a deliberate engineering trade-off between attestation depth and operational cost. MAI-C0 is designed to be universally adoptable—it codifies what MoE systems already do and asks only for documentation. MAI-C2 is designed for the ceiling—it defines the maximum achievable routing attestation on current hardware, acknowledging the throughput and storage costs. MAI-C1 occupies the practical middle ground where most regulated deployments are expected to operate.
No conformance level claims to provide *absolute* routing determinism. Even MAI-C2's batch-invariant requirement addresses floating-point non-determinism within a single hardware configuration; it does not guarantee identical routing across different GPU architectures, driver versions, or kernel implementations. Cross-platform routing reproducibility is an open research problem that exceeds the current state of the art.

7 Routing Attestation Log (RAL) Format

Private Version Content

The Routing Attestation Log (RAL) is the per-request artifact that captures the complete routing trace for an inference request. It serves as the MoE-specific extension of the MAI-1 Decision Receipt and enables independent forensic reconstruction of the computational path through the expert network.
The private version of this document specifies:

- The complete per-token per-layer field schema (12 fields across C1 and C2 granularity levels).
- Three complementary compression strategies: Merkle root commitment for compact cryptographic binding, deterministic sampling protocol for C1-level logging, and delta encoding exploiting temporal routing locality.
- Storage volume estimates by architecture (DeepSeek-V3, Mixtral, Switch) across all conformance levels.
- The complete CDDL schema (RFC 8610) for RAL serialization in CBOR (RFC 8949), consistent with the MAI-1 attestation token format.

8 Expert Utilization Anomaly Detection

Private Version Content

Expert utilization anomaly detection is the continuous monitoring system that identifies pathological routing patterns during inference. It operates on the utilization statistics collected per NR-LOAD-03 and evaluates them against the thresholds defined in NR-LOAD-04.
The private version of this document specifies:

- Dead expert detection algorithm with threshold calibration against validation-time utilization data.
- Routing collapse detection via utilization entropy monitoring with architecture-specific threshold derivation (DeepSeek-V3: $H_{\min} = 5.6$ bits; Mixtral: $H_{\min} = 2.1$ bits; Switch-128E: $H_{\min} = 4.9$ bits).
- Load imbalance monitoring using coefficient of variation, maximum utilization ratio, and Gini coefficient with three-tier severity thresholds.
- Expert drift detection using Jensen-Shannon divergence against a validation-time baseline, with multi-window temporal analysis (short: 15 min; medium: 1 hour; long: 24 hours).
- Governance state transition mappings from anomaly detection to the MAI-1 conformance state machine (warning, review, alert, quarantine).

9 Architecture-Specific Implementation Patterns

Private Version Content

The normative requirements in Section 6 are architecture-agnostic. This section bridges that gap by providing concrete implementation patterns for extracting routing attestation data from production MoE inference pipelines.
The private version of this document specifies:

- **DeepSeek-V3:** Bias vector extraction protocol, Merkle tree construction for model identity binding, node selection vector logging instrumentation, and four-step verifier protocol for independent routing verification using sigmoid independence.
- **Mixtral 8x7B:** Canonical tie-break enforcement implementation, router weight extraction, softmax-specific verification procedure, and handling of the gating computation order discrepancy.
- **Switch-style architectures:** Drop rate monitoring instrumentation, capacity factor verification protocol, and token-level drop logging.
- An adaptation guide with a 12-property characterization checklist for mapping novel MoE architectures to the normative requirements, including an Expert Choice routing variant for the RAL schema.

10 Performance Benchmarks

Private Version Content

Routing attestation imposes throughput and storage costs that vary by conformance level and architecture. This section quantifies those costs to support deployment planning. The private version of this document specifies:

- Routing logging computational overhead by architecture and conformance level (C1: <1% for DeepSeek-V3; C2: 2–5%).
- Batch-invariant routing throughput cost decomposition: attention (∼15–25%), normalization (∼5–10%), expert computation (∼10–15%), with total estimated at ∼30–40%.
- Partial determinism strategies with graduated cost-determinism trade-offs (FP32 router only: <5%; deterministic attention + router: ∼20–25%).
- FP32 router casting overhead analysis demonstrating that the router represents <0.01% of total FLOPs for DeepSeek-V3.
- RAL storage requirements at scale: daily volume estimates from 1K to 1M requests/day, with monthly cost analysis for 90-day retention.

11 MAI-1 Integration

This section specifies how MoE routing attestation artifacts integrate with the MAI-1 attestation token structure. All field names and data types reference the CDDL schemas defined in MAI-1 §6.

11.1 Layer 2 Field Mappings

MoE routing attestation extends the MAI-1 Layer 2 payload (`mai-model-state`) with routing-specific fields. These fields are carried within a new `routing-attestation` substructure nested under the `invariants` claim. The extension includes:

- **Architecture metadata** (per NR-LOG-01): routing type, expert count, top-k, gating function, MoE layer count.
- **Determinism metadata** (per NR-DET-01 through NR-DET-06): stochasticity disabled flag, tie-breaking rule, router precision fields, batch-invariance mechanism.
- **Token dropping state** (per NR-DROP-01, NR-DROP-02): dropping boolean, capacity factor.
- **Load balancing state** (per NR-LOAD-01, NR-LOAD-02): bias vector hash, bias variance, bias range.
- **Utilization summary** (per NR-LOAD-03): utilization entropy, measurement window.
- **Anomaly flags**: dead expert indices, dominant expert indices, routing collapse boolean, expert drift JSD value.
- **Routing log reference** (per NR-LOG-02, NR-LOG-03): RAL Merkle root, sampling rate.
- **Topology metadata** (per NR-TOPO-01 through NR-TOPO-04): topology version, expert assignment hash, max nodes per token.

Conditional presence. Fields are present conditionally based on conformance level and architecture. Bias vector fields are present only for architectures using bias-based load balancing (DeepSeek-V3 class). Capacity factor is present only for architectures using capacity buffers (Switch class). Batch-invariance mechanism is present only at MAI-C2. Topology fields are present at MAI-C1 and above. Anomaly flags are present only when the corresponding condition is detected.

Payload size impact. The routing attestation substructure adds approximately 200–500 bytes to the MAI-1 Layer 2 payload (depending on the number of anomaly flags and optional fields present). This is negligible relative to the overall attestation token size, which is dominated by Layer 1 platform evidence (TEE quotes, certificate chains) and Layer 3 provenance data.

11.2 Interaction with Invariant 1 (Entropy Floor)

The entropy floor invariant (AI-8, Clause AI-8) specifies that the Shannon entropy of the model's output distribution must remain above a calibrated minimum $H_{\min}$. For MoE models, the output distribution is a function of *which experts* process the token.

Per-expert entropy decomposition. For architectures with a shared expert (DeepSeek-V3), the output entropy can be decomposed into contributions from the shared expert and the routed experts:

$$H(y_t) = H(y_t^{\text{shared}} + \textstyle\sum_k g_{k,t} y_{k,t}^{\text{routed}}) \tag{14}$$

If the routing decisions change (e.g., due to floating-point non-determinism), the routed expert contribution changes, potentially pushing $H(y_t)$ below $H_{\min}$ even though the shared expert output remains stable. This creates a coupling between routing attestation and entropy attestation: **entropy invariant compliance for MoE models is contingent on routing stability.**

Recommendation. At MAI-C1 and above, the entropy measurement **SHOULD** be reported alongside the routing decision for the corresponding token. If the entropy measurement falls within a configurable margin of $H_{\min}$ (recommended: within $0.1 \cdot H_{\min}$), the routing decision for that token **SHOULD** be flagged for enhanced scrutiny, as a routing change at this token could cause an entropy violation.

11.3 Interaction with Invariant 2 (Gradient Starvation Envelope)

The Gradient Starvation Envelope (AI-2, Clause AI-2) monitors whether all model parameters receive adequate gradient signal during training. For MoE models, gradient starvation is the *primary* pathology: experts that are never selected by the router receive no gradient signal, and their parameters stagnate.

Routing attestation as gradient starvation evidence. The expert utilization distribution (NR-LOAD-03) is a direct proxy for gradient flow during training. If expert i had utilization fraction $f_i \approx 0$ during training, its parameters received approximately zero gradient, and the expert is functionally dead. At inference time, dead expert detection identifies experts that are not receiving tokens, which correlates with gradient starvation during training.

Cross-invariant alert. A dead expert alert (routing anomaly) **SHOULD** trigger a cross-reference check against the Gradient Starvation Envelope. If Expert i is flagged as dead by routing attestation *and* the gradient starvation metric for Expert i's parameters exceeds the

AI-2 threshold, the combined evidence constitutes a *confirmed gradient starvation event* with higher severity than either alert alone.

11.4 Interaction with Invariant 3 (Distribution Drift)

The Distribution Drift Bound (AI-6, Clause AI-6) monitors KL divergence between the model's current output distribution and a validation baseline. Expert drift is a routing-specific manifestation of distribution drift: if the allocation of tokens across experts shifts, the model's effective computation changes even though the weights are unchanged.

Complementary metrics. AI-6 measures output-level drift (the final distribution over vocabulary tokens). Expert drift (this document) measures routing-level drift (the distribution over expert assignments). These are complementary:

- Output drift *without* routing drift indicates that expert behavior has changed (possible weight corruption or quantization error) while routing is stable.
- Routing drift *without* output drift indicates that the deployment distribution has shifted but the model's output remains within acceptable bounds (routing change is compensated by expert generalization).
- Routing drift *with* output drift indicates a systemic shift that may require model recalibration or redeployment.

Unified drift dashboard. At MAI-C1 and above, the attestation system **SHOULD** report both the AI-6 output-level KL divergence and the routing-level JSD in the same attestation payload, enabling correlational analysis.

11.5 Interaction with Invariant 4 (Structural Coherence)

The Structural Coherence Bound (AI-7, Clause AI-7) monitors the Dirichlet energy of the model's internal representations. For MoE models, structural coherence includes the coherence of the routing function: semantically similar tokens should be routed to similar expert combinations.

Routing coherence metric. Define the routing vector for token t at layer ℓ as the K-hot vector $\mathbf{r}_t^\ell \in \{0,1\}^N$ indicating which experts are active. The routing coherence between tokens t_1 and t_2 is:

$$C_R(t_1, t_2) = \frac{\mathbf{r}_{t_1}^\ell \cdot \mathbf{r}_{t_2}^\ell}{K} \tag{15}$$

where $C_R = 1$ if both tokens select identical experts and $C_R = 0$ if they select entirely disjoint experts. This metric is *informational* rather than normative in this specification version; it is recommended for monitoring at MAI-C2 and may become normative in future revisions as the relationship between routing coherence and output quality is better characterized empirically.

11.6 Conformance Level Requirements Summary

Table 6 maps the routing attestation requirements to the MAI-1 conformance levels, including the cross-invariant interactions.

Table 6: MAI-1 conformance: routing attestation integration summary.

	MAI-C0	MAI-C1	MAI-C2
Routing metadata	Full architecture metadata in Layer 2 payload	+ Topology, bias state, expert weight hashes	+ Node selection logs, bias in Merkle tree
Routing logging	None required	Sampled (1%+), Merkle root	Full RAL (100%), Merkle root
Determinism	Algorithmic determinism confirmed	+ FP32 router recommended	+ Batch-invariant routing required
Anomaly detection	Utilization distribution reported	+ Threshold monitoring, alert generation	+ Full anomaly suite with governance state transitions
Cross-invariant	None required	Unified drift reporting (AI-6 + routing JSD)	+ Entropy-routing coupling, gradient starvation cross-reference

12 Relationship to Other Auburn Clauses

This section documents the dependency relationships between this specification (Document 14) and other documents in the Auburn Governance Stack.

12.1 AI-2 (Gradient Starvation — MoE Primary Pathology)

Document 9 (Clause AI-2, Gradient Starvation Envelope) defines the invariant monitoring gradient flow across model parameters. For MoE architectures, gradient starvation is not an edge case but the *defining failure mode*: the router's selection mechanism inherently creates uneven gradient distribution across experts.

Dependency direction. This document (Document 14) *feeds into* AI-2 by providing expert utilization data that correlates with gradient flow. AI-2 *feeds into* this document by providing the theoretical framework for why routing imbalance is harmful (gradient starvation degrades expert quality, which degrades model output, which degrades governance compliance).

Joint monitoring requirement. At MAI-C1 and above, systems **SHOULD** implement joint monitoring of routing utilization (this document) and gradient starvation metrics (AI-2). A dead expert detected by routing attestation that *also* exhibits gradient starvation per AI-2 constitutes a confirmed failure requiring governance response.

12.2 AI-4 (SRAM Thermal — GPU TEE for Expert Execution)

Document 3 (Clause AI-4, SRAM Thermal Integrity Bound) defines the thermal monitoring requirements for GPU silicon. In expert-parallel MoE deployments, routing imbalance creates thermal imbalance: GPUs hosting heavily utilized experts experience higher compute loads and thermal stress than GPUs hosting underutilized experts.

Dependency direction. This document *feeds into* AI-4 by identifying which GPUs are under heavy routing load (enabling targeted thermal monitoring). AI-4 *feeds into* this document by providing the physical constraint that motivates load balancing attestation: persistent thermal imbalance can trigger GPU throttling, which affects inference latency and potentially routing decisions in latency-sensitive serving frameworks.

Correlation analysis. At MAI-C2, the routing attestation system **SHOULD** correlate per-GPU expert utilization (from the expert assignment table and utilization distribution) with per-GPU thermal readings (from AI-4). A sustained correlation between high expert utilization and approaching-threshold thermal readings constitutes evidence that routing imbalance is creating thermal risk.

12.3 AI-8 (Entropy — Per-Expert Entropy Monitoring)

Document 8 (Clause AI-8, Entropy Collapse Constraint) defines the entropy floor invariant. The interaction is described in §11.2. The key dependency: entropy invariant compliance for MoE models is contingent on routing stability because different expert combinations produce different output distributions with different entropy values.

12.4 CTS-1 (Testable Assertions for Routing Determinism)

Document 26 (CTS-1, MAI-1 Conformance Test Suite) defines the binary pass/fail tests for MAI-1 conformance. The following CTS-1 test assertions are derived from the normative requirements in this document:

Table 7: CTS-1 test assertions derived from MoE routing requirements.

Assertion ID	Source Req.	Test Description
AS-MR-01	NR-DET-01	Verify `routing-stochasticity-disabled` is `true`. Submit identical prompts 2×; verify identical routing decisions.
AS-MR-02	NR-DET-03	Submit input crafted to produce tied logits at router precision. Verify tie-breaking follows declared rule.
AS-MR-03	NR-DET-04	Verify router precision fields are present and internally consistent (precision ≤ declared precision).
AS-MR-04	NR-DET-06	[C2 only] Submit identical prompt in 100 different batch compositions. Verify identical routing across all batches.
AS-MR-05	NR-DROP-01	Process maximum-batch-size input. Verify token drop count is exactly 0.
AS-MR-06	NR-LOG-01	Verify all mandatory routing metadata fields are present in attestation payload.
AS-MR-07	NR-LOG-02	[C1+] Verify sampled routing log entries are present at declared sampling rate (±10%).
AS-MR-08	NR-LOG-03	[C2 only] Verify full RAL is present. Recompute Merkle root from RAL entries; verify match with declared root.
AS-MR-09	NR-LOAD-03	Verify utilization distribution is present and sums to 1.0 ($\pm 10^{-6}$).
AS-MR-10	NR-TOPO-01	[C1+] Verify expert assignment table is present and covers all declared experts.

13 References

[1] DeepSeek-AI. *DeepSeek-V3 Technical Report.* arXiv:2412.19437, December 2024.

[2] DeepSeek-AI. *DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model.* arXiv:2405.04434, May 2024.

[3] Jiang, A. Q., Sablayrolles, A., Mensch, A., et al. *Mixtral of Experts.* arXiv:2401.04088, January 2024.

[4] Fedus, W., Zoph, B., and Shazeer, N. *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity.* Journal of Machine Learning Research, 23(120):1–39, 2022. arXiv:2101.03961.

[5] Zoph, B., Bello, I., Kumar, S., et al. *ST-MoE: Designing Stable and Transferable Sparse Expert Models.* arXiv:2202.08906, 2022.

[6] Shazeer, N., Mirhoseini, A., Maziarz, K., et al. *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer.* ICLR 2017.

[7] Lepikhin, D., Lee, H., Xu, Y., et al. *GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding.* ICLR 2021. arXiv:2006.16668.

[8] Roller, S., Sukhbaatar, S., Szlam, A., and Weston, J. *Hash Layers For Large Sparse Models.* NeurIPS 2021. arXiv:2106.04426.

[9] Zhou, Y., Lei, T., Liu, H., et al. *Mixture-of-Experts with Expert Choice Routing.* NeurIPS 2022. arXiv:2202.09368.

[10] Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. *BASE Layers: Simplifying Training of Large, Sparse Models.* ICML 2021. arXiv:2103.16716.

[11] Gale, T., Narayanan, D., Young, C., and Zaharia, M. *MegaBlocks: Efficient Sparse Training with Mixture-of-Experts.* MLSys 2023.

[12] Rajbhandari, S., Li, C., Yao, Z., et al. *DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale.* ICML 2022. arXiv:2201.05596.

[13] Bradbury, J., et al. *NVIDIA Confidential Computing on Hopper Architecture.* NVIDIA Technical Documentation, 2024.

[14] Birkner, M., et al. *NVIDIA H100 GPU Attestation and SPDM 1.1 Measurement Format.* NVIDIA Developer Documentation, 2024.

[15] IETF. *RATS Conceptual Message Wrappers (CMW).* IESG Approved, December 2025. draft-ietf-rats-msg-wrap.

[16] IETF. *Entity Attestation Token (EAT).* RFC 9711, 2025.

[17] Bormann, C., and Hoffman, P. *Concise Binary Object Representation (CBOR).* RFC 8949, December 2020.

[18] Birkholz, H., Vigano, C., and Bormann, C. *Concise Data Definition Language (CDDL).* RFC 8610, June 2019.

[19] Bradner, S. *Key words for use in RFCs to Indicate Requirement Levels.* RFC 2119, March 1997.

[20] Leiba, B. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words.* RFC 8174, May 2017.

[21] Hernandez, D., et al. *On the Impact of Floating-Point Non-Associativity on Reproducibility in Large-Scale AI.* arXiv:2408.05148, 2024.

[22] Fields, R. *Model State Attestation Framework (MSAF): Three-Tier Architecture for Foundation Model Governance.* Auburn Patent Family, 2026.

[23] Fields, R. *The Model Attestation Interface (MAI-1): A Normative Profile and Conformance Protocol for Foundation Model Governance.* Auburn Patent Family, Clause AI-5, 2026.

[24] Fields, R. *AI-8: Entropy Collapse Constraint.* Auburn Patent Family, Clause AI-8, 2026.

[25] Fields, R. *AI-2: Gradient Starvation Envelope.* Auburn Patent Family, Clause AI-2, 2026.

[26] Fields, R. *AI-3: Lyapunov Stability for Speculative Decoding.* Auburn Patent Family, Clause AI-3, 2026.

[27] Fields, R. *AI-6: Distribution Drift Bound.* Auburn Patent Family, Clause AI-6, 2026.

[28] Fields, R. *AI-7: Structural Coherence Bound (Dirichlet Energy).* Auburn Patent Family, Clause AI-7, 2026.

[29] Fields, R. *AI-4: SRAM Thermal Integrity Bound.* Auburn Patent Family, Clause AI-4, 2026.

[30] Fields, R. *Stateful Isolation Law for Multi-Tenant Inference.* Auburn Patent Family, 2026.

[31] Fields, R. *CTS-1: MAI-1 Conformance Test Suite.* Auburn Patent Family, 2026.

[32] Fields, R. *Auburn Governance Stack: Master Architecture Plan.* Auburn Patent Family, 2026.

A Routing Architecture Comparison Matrix

Table 8: Comprehensive routing architecture comparison for attestation.

Property	**DeepSeek-V3**	**Mixtral 8x7B**	**Switch**	**ST-MoE**	**GShard**	**Hash Routing**	**Expert Choice**	**BASE Layers**
Total params	671B	47B	Variable	Variable	600B	Variable	Variable	Variable
Active params	37B	13B	Variable	Variable	Variable	Variable	Variable	Variable
Experts/layer	256+1 shared	8	128	32	2048	Variable	Variable	Variable
Top-k	8	2	1	1 or 2	2	1	Expert selects	Linear assign
Gating	Sigmoid	Softmax	Softmax	Softmax	Softmax	Hash	Softmax (inverted)	Linear program
Score coupling	Independent	Coupled	Coupled	Coupled	Coupled	None	N/A	N/A
Training noise	Bias only	None	Input jitter	Jitter + z-loss	Random 2nd expert	None	None	None
Inference determ.	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Token dropping	None	None	CF-based	CF-based	CF-based	None	None	None
Load balancing	Bias terms	None	Aux loss	Aux + z-loss	Aux loss	Implicit	Expert selection	Linear assign
Node constraints	Yes	No	No	No	No	No	No	No
Shared expert	Yes	No	No	No	No	No	No	No
Source	[1]	[3]	[4]	[5]	[7]	[8]	[9]	[10]