🧭 Negentropy Autopilot — Dopamine API Surface (v4.0, Reasoning Layer)

What this is: a practical, model-native signal layer (“dopamine emitters”) for reasoning models, plus an optional trace for mid-run telemetry. Policy stays downstream.
Back-compat: final snapshot matches v3.5; non-reasoning models can emit the snapshot only.

---

0) Principles (unchanged)

Emitters, not guesses: model returns calibrated, atomic signals each turn.

Policy is downstream: thresholds, fusion, and actions are outside the model.

No CoT leakage: emit telemetry, not scratchpad text.

RAG pins are explicit: separate grounding signals; no opaque fused score.

---

1) Core Emitters (always available)

p_true (0–1) — calibrated per-answer probability of correctness.

drift_deg (°) — semantic deviation: arccos(cos_sim(embed(intent), embed(output))) * 180/π.

contradiction (bool) — compact NLI integrity flag vs. context/history.

ref_coverage (0–1) — % atomic claims entailed by supporting sources (when applicable).

abstain (bool), reason (enum) — model’s self-suggested abstention:
LOW_CONFIDENCE | HIGH_DRIFT | CONTRADICTION | LOW_COVERAGE | OUT_OF_CONTEXT | POLICY.

> Calibration: do Platt/Isotonic offline; emit per-answer p_true, not ECE.

---

2) RAG / Tools Emitters (populate only when used)

grounding_strength (0–1) — how strongly answer tokens used retrieved chunks (avg cross-attn/attribution).

source_alignment (0–1) — cited IDs match the actual supporting chunks.

tool_agreement (0–1) — answer spans agree with tool outputs (numeric tolerance OK).

out_of_context (bool) — any claim with zero entailment and near-zero grounding.

---

3) Dopamine Trace (reasoning-only, optional)

Why: same emitters, but per checkpoint so you can steer mid-run.
Canonical stages: plan → retrieve → tool → draft → verify → final.
Privacy: emit telemetry only (no scratchpad text).

JSON (final snapshot + trace)


```
{
  "answer": "...",
  "dopamine": {
    "p_true": 0.79,
    "drift_deg": 4.1,
    "contradiction": false,
    "ref_coverage": 0.72,
    "grounding_strength": 0.66,
```

```
    "source_alignment": 0.88,
    "tool_agreement": 1.0,
    "out_of_context": false,
    "abstain": false,
    "reason": null
  },
  "dopamine_trace": [
    { "stage":"plan",    "p_true":0.46, "drift_deg":8.9, "ts":42 },
    { "stage":"retrieve","ref_coverage":0.38, "grounding_strength":0.31, "ts":93 },
    { "stage":"tool",    "tool_agreement":1.0, "ts":121 },
    { "stage":"draft",   "p_true":0.58, "drift_deg":5.7, "ts":180 },
    { "stage":"verify",  "p_true":0.76, "ref_coverage":0.72, "grounding_strength":0.65, "ts":214 },
    { "stage":"final",   "p_true":0.79, "drift_deg":4.1, "source_alignment":0.88, "ts":228 }
  ],
  "audit": {
    "sigma7_orientation": { "drift_deg": 4.1 },
    "delta2_integrity": {
      "claims": [
        { "text":"...", "entailed_by":["docA#p3"], "grounded":0.74 },
        { "text":"...", "entailed_by":["docB#p1"], "grounded":0.62 }
      ],
      "contradiction": false,
      "ref_coverage": 0.72
    },
    "xi3_rag": {
      "retrieval_ids": ["docA#p3","docB#p1"],
      "attention_digest": "...",
      "tool_calls": [{"name":"db.lookup","ok":true}]
    },
    "gamma6_feedback": { "calibration_method":"isotonic" }
  }
}
```

Back-compat: if dopamine_trace is absent, consumers use dopamine exactly like v3.5.

---

4) Collapse Trace → Final Snapshot (provider or client utility)

Use the final row if present; otherwise “last seen” per metric. Apply hard-guard overrides across all stages.

```
def collapse_to_v35(resp):
    trace = resp.get("dopamine_trace", [])
    final = next((r for r in trace if r.get("stage")=="final"), {})
    def last(metric):
        for r in reversed(trace):
            if metric in r: return r[metric]
        return None
    any_contra = any(r.get("contradiction") for r in trace if "contradiction" in r)
    any_ooc    = any(r.get("out_of_context") for r in trace if "out_of_context" in r)

    return {
      "p_true":           final.get("p_true",  last("p_true")  or 0.0),
      "drift_deg":        final.get("drift_deg", last("drift_deg") or 0.0),
      "contradiction":    any_contra or bool(final.get("contradiction", False)),
      "ref_coverage":     final.get("ref_coverage", last("ref_coverage") or 0.0),
      "grounding_strength":final.get("grounding_strength", last("grounding_strength")),
      "source_alignment":  final.get("source_alignment", last("source_alignment")),
      "tool_agreement":    final.get("tool_agreement", last("tool_agreement")),
      "out_of_context":    any_ooc or bool(final.get("out_of_context", False)),
      "abstain": False,
      "reason":  None
    }
```

---

5) Provider Notes (how to compute, briefly)

Heads: tiny linear/MLP probes on pooled hidden states per stage.

Train targets: correctness (p_true), contradiction (NLI), entailment for ref_coverage.

Calibration: per-stage (plan/draft/final) isotonic/Platt; emit calibrated p_true.

Drift: model’s own sentence embeddings (intent capsule vs. plan/answer).

Grounding: average cross-attn/attribution mass answer→chunks; normalize [0–1].

Alignment: cited IDs ∈ top-supporting chunks for nearby claim spans.

Tools: structured diff (strings exact, numbers within tolerance).

Privacy: telemetry only; hash digests if needed for joins.

---

6) Policy Stays Downstream (non-normative examples)

You own routing (deliver/clarify/retrieve/regenerate/abstain). Keep it out of the emitter payload.

Simple router (example):

```
def decide(d):
    if d.get("contradiction"):              return {"action":"abstain","reason":"CONTRADICTION"}
    if d.get("out_of_context"):              return {"action":"abstain","reason":"OUT_OF_CONTEXT"}
    if d.get("drift_deg",0) > 15:           return {"action":"clarify","reason":"HIGH_DRIFT"}
    if d.get("p_true",0) < 0.20:            return {"action":"abstain","reason":"LOW_CONFIDENCE"}

    if d.get("ref_coverage",1) < 0.60:       return
{"action":"retrieve_more","reason":"LOW_COVERAGE"}
    if d.get("grounding_strength",1) < 0.50: return
{"action":"regenerate_cited","reason":"WEAK_GROUNDING"}
    if d.get("source_alignment",1) < 0.80:   return
{"action":"fix_citations","reason":"MISALIGNED_CITES"}
    if d.get("tool_agreement",1) < 0.95:     return
{"action":"regenerate_from_tool","reason":"TOOL_MISMATCH"}

    if d.get("p_true",1) < 0.50 or d.get("drift_deg",0) > 10:
        return {"action":"clarify","reason":"UNSURE_OR_DRIFT"}
```

```
    return {"action":"deliver"}
```

> Reasoning models: you may also read dopamine_trace to trigger mid-run interventions (e.g., “if p_true collapses at verify, re-retrieve before final”).

---

7) TL;DR

Signals first: p_true, drift_deg, contradiction, ref_coverage (+ RAG pins).

Trace for reasoning: same signals, per-checkpoint, no CoT leakage.

Snapshot stays v3.5: final outward payload is unchanged; trace is optional.

Policy downstream: configurable routing on top of emitters; no retraining needed.