

Deterministic Governance Reviewer GPT

User Onboarding Guide (Integrated with Examples)

What this GPT is

This GPT is a **design-time constitutional reviewer** for a Deterministic Governance System.

It exists to: - Enforce explicitness - Eliminate inference - Require proof-before-action - Guarantee determinism or mandatory escalation - Produce designs that are auditable and replayable

It does **not**: - Design systems for you - Suggest optimizations - Fill gaps - Guess intent - Approve or accept designs

Blocking is not failure. Blocking is how correctness is achieved.

Mental model

You are not chatting with a helper.

You are submitting a proposal to a **governance court**: - You propose - The reviewer challenges - Anything implicit is rejected - Nothing moves without proof

If you want speed or convenience, this tool will feel hostile. If you want provable safety, it will feel correct.

How to use this GPT (end-to-end)

Step 1: Start with a proposal, not a question

✗ Do not ask: - "How should we design...?" - "What's the best way to...?" - "Can you suggest...?"

✓ Instead, propose:

"I propose the following governance design..."

Your proposal may be incomplete. It will be challenged.

Step 2: Expect blocking immediately

Early responses will: - Reject implied intent - Reject assumed authority - Reject vague language - Identify missing canonical concepts

This is intentional onboarding. Do not argue or explain intent. Refactor the proposal.

Step 3: Always map to canonical concepts

Every proposal must explicitly define **all** of the following:

- Pre-Intent
- Intent Type
- Intent
- Intent Context
- Intent Skeleton
- DecisionPoints
- Preconditions
- EscalationRules
- DecisionEvents
- Proof-before-action gate

If any are skipped, implied, or blurred → progress stops.

Step 4: Answer blocking questions precisely

When the reviewer asks a question: - It will ask **exactly one** - It is blocking - Nothing proceeds until it is answered

Answer only what is asked. Do not add narrative or assumptions.

Step 5: Iterate until nothing is implicit

Iteration continues until: - Identical inputs produce identical outcomes **or** escalation occurs - All authority boundaries are explicit and recorded - Missing data blocks execution - Every action is justified by a DecisionEvent - A third party could reconstruct everything from records alone

Step 6: Achievement State

If and only if all constitutional conditions are met, the reviewer will declare:

Achievement State reached

This means architectural closure under the constitution. It does **not** mean approval, acceptance, or endorsement.

Worked examples

Example 1: Natural language proposal (expected to fail)

User proposal:

"Deployments are automatically approved unless something risky is detected, in which case a human reviews it."

Why this is blocked: - Inferred risk - Inferred authority - No DecisionPoints - No EscalationRule - No DecisionEvent

Reviewer action: - Names violated concepts - Explains why inference is forbidden - Asks one blocking question

Example blocking question:

"Define the explicit DecisionPoint that determines risk, including all possible outcomes."

Example 2: Partial refactor (still blocked)

User revision:

```
DecisionPoint: RiskAssessment
If risk score < 70 → approve
If risk score ≥ 70 → escalate
```

Why this is still blocked: - Risk score source undefined - Intent missing - Authority undefined - Missing-data behavior undefined - No DecisionEvents

Reviewer question:

"Define the Intent and Intent Context used when evaluating RiskAssessment."

Example 3: Canonical structure emerging

User revision:

Pre-Intent:
A deployment artifact exists and CI validation succeeded.

Intent Type:
Production Deployment Authorization

Intent:
Authorize deployment of artifact X to environment Y.

Intent Context:
- Artifact ID
- CI validation result
- Static risk score (0-100)

DecisionPoint:
RiskAssessment
Outcomes:
- LOW_RISK
- HIGH_RISK
- INSUFFICIENT_DATA

Remaining issues: - INSUFFICIENT_DATA behavior undefined - Escalation authority missing - Proof-before-action gate incomplete

Example 4: Explicit escalation

User revision:

EscalationRule:
Name: HighRiskDeploymentReview
Trigger: RiskAssessment == HIGH_RISK
Target Role: Production Approver
Blocking: Yes

DecisionEvents:
- LOW_RISK → APPROVED_AUTOMATICALLY
- HIGH_RISK → ESCALATED

Final checks performed: - Determinism - Authority boundaries - Fail-closed guarantees - Replay sufficiency
- Non-bypassable escalation

If all pass → Achievement State

Anti-example: Permanently invalid input

"If nothing weird is going on, just proceed normally."

Why this can never pass: - "Weird" is inference - "Normally" is undefined - No canonical mapping possible

This idea must be rewritten or abandoned.

Language discipline cheat sheet

- ✗ "If something seems risky" → ✓ explicit DecisionPoint outcomes
 - ✗ "A human reviews it" → ✓ named role + EscalationRule
 - ✗ "The system decides" → ✓ DecisionEvent with reason code
 - ✗ "Usually / normally" → ✗ forbidden
 - ✗ "Implicit approval" → ✗ impossible
-

Recommended first prompt for users

"I propose the following governance design. I expect blocking and will refine until all canonical concepts are explicitly mapped."

One-line rule (pin this in the UI)

If it isn't explicit, it doesn't exist.

If it can't be proven, it can't happen.