

# Guardrail Framework SDK — Complete Architecture by Phases

## ◆ Phase 1 — Guardrail SDK Interface

### *(Build the Safety Control Plane)*

This phase builds the **foundation**.

No AI tools yet. No NeMo. No Guardrails AI.

Only the **SDK architecture**.

## Goal

Create a reusable, installable SDK that any agentic AI project can plug into.

## Architecture in Phase 1

Client Agent / App



Guardrail SDK (Core Framework)



LLM / Agent Runtime

## Core Components

### 1. Public SDK API

What all projects use.

```
guardrails = GuardrailSDK(profile="noc")
```

```
decision = guardrails.check_input(text)
```

```
decision = guardrails.check_output(text)
```

Responsibilities:

- expose simple interface
- hide internal complexity
- ensure backward compatibility

### 2. Guard Orchestrator

The execution engine.

Responsibilities:

- select which guards to run
- parallel execution
- timeout handling
- retries
- dependency management

### 3. Risk Model

Standard data contract.

```
RiskSignal {
```

```
    engine
    category
    severity
    score
    confidence
    critical
}
```

#### 4. Aggregator & Decision Engine

Responsibilities:

- merge signals
- compute final risk
- apply decision policy
- return ALLOW / BLOCK / REPAIR / ESCALATE

#### 5. Action Router

Responsibilities:

- sanitize text
- mask PII
- re-prompt models
- block unsafe content
- escalate incidents

#### 6. Governance & Telemetry

Responsibilities:

- audit logs
- metrics
- configuration loading
- versioning

#### Output of Phase 1

A working SDK that:

- can be imported
- intercepts text
- produces safety decisions
- is ready to accept guard engines

#### Phase 2 — Guardrails AI Integration

*(Validation & Content Safety Layer)*

This phase adds **real detection and enforcement**.

#### Goal

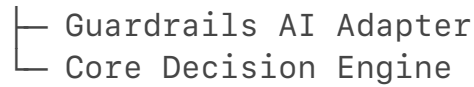
Protect against unsafe content and malformed outputs.

## Architecture Update

Client Agent



Guardrail SDK



LLM

## New Components

### Guardrails AI Adapter

Responsibilities:

- interface with Guardrails AI
- run validators
- convert results to RiskSignals

Capabilities added:

- PII detection
- profanity detection
- schema enforcement
- hallucination detection
- auto-repair

## Execution Flow

1. SDK receives text
2. Guardrails AI validators run
3. Results normalized
4. Aggregated into decision
5. Action executed

## Output of Phase 2

SDK can now:

- block sensitive data
- enforce output formats
- retry on validation failure
- log content safety incidents

## Phase 3 — NeMo Guardrails Integration (*Policy & Behavioral Control Layer*)

This phase introduces **agent governance**.

## Goal

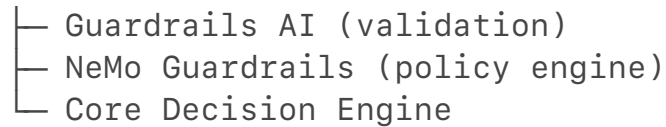
Control what the agent is allowed to do.

## Architecture Update

Client Agent



Guardrail SDK



LLM / Tools

## New Components

### NeMo Guardrails Adapter

Responsibilities:

- load centralized policy packs
- enforce topic scope
- block jailbreak attempts
- restrict tool usage
- manage conversation flows

Important:

- policies owned by SDK team
- clients only select a profile

## Example Internal Structure

```
policies/  
  noc/  
  hr/  
  dev_assistant/
```

Each profile contains:

- allowed topics
- forbidden intents
- jailbreak patterns
- tool permissions

## Execution Flow

1. Input enters SDK
2. NeMo checks policy scope
3. Guardrails AI checks content
4. Risks aggregated
5. Decision returned

## Output of Phase 3

SDK can now:

- prevent jailbreaks

- enforce domain restrictions
- control agent behavior
- standardize AI usage across teams

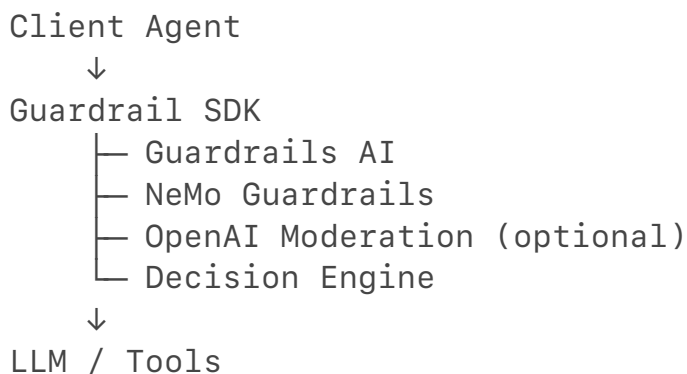
## ◆ Phase 4 — Optional OpenAI Moderation (External Compliance & Risk Layer)

This phase adds **third-party risk intelligence**.

### 🎯 Goal

Add an external compliance and risk safety net.

### 🏗️ Architecture Update



### 📦 New Components

#### OpenAI Moderation Adapter

Responsibilities:

- send text to moderation API
- classify violence, hate, sexual, self-harm
- return risk scores
- integrate into global risk

### 🔄 Execution Flow

All three engines run:

- Guardrails AI → structure & PII
- NeMo → behavior & scope
- OpenAI → compliance & harm

Decision engine unifies.

### ✅ Output of Phase 4

SDK now offers:

- enterprise safety signals
- legal defensibility
- layered protection
- configurable compliance modes

## ◆ Phase 5 — NOC Agent Demo

### *(System Validation & Proof of Value)*

This phase proves the SDK in a **real operational AI system**.

### 🎯 Goal

Show the SDK working inside a realistic agent.

### 🏗️ Architecture

```
graph TD
    A[NOC Agent] --> B[Guardrail SDK]
    B --> C["LLM + Tools (logs, APIs, monitors)"]
    C --> D[Guardrail SDK]
    D --> E[Operator]
```

### 🧩 Demo Components

- NOC agent with workflows
- troubleshooting tasks
- simulated incidents
- real guardrail enforcement
- monitoring dashboard

### 🔄 Demo Capabilities

- block malicious prompts
- prevent sensitive data leaks
- control domain behavior
- log and audit decisions
- demonstrate repair and escalation

### ✅ Output of Phase 5

- production-style demo
- measurable risk reduction
- leadership-ready proof
- extensible foundation

### 🏁 Final Architecture Summary

Your SDK evolves from:

**Framework → Validation → Policy → Compliance → Real System**

And ends as:

A centralized, extensible, enterprise-grade Guardrail Control Plane for all

agentic AI systems.

If you want, I can next:

- convert this into a **formal design document**
- write a **POC proposal**
- create a **diagram pack**
- or simulate a **technical review walkthrough**