# AWS Bedrock Guardrails × EthicalZen.ai + SentryWorks.ai Integration Modes

## Executive summary

This document describes practical ways to integrate AWS Bedrock Guardrails with EthicalZen.ai and SentryWorks.ai. The goal is to give customers multiple adoption paths: from augmenting Bedrock's native controls, to using EthicalZen as an end-to-end safety layer, to making SentryWorks the policy control plane across multiple LLM providers.

### At a glance

- Mode A (Complement): keep Bedrock Guardrails; add EthicalZen Smart Guardrails that Bedrock does not provide.
- Mode B (Orchestration): design guardrails in EthicalZen; provision/manage Bedrock Guardrails via API for Bedrock apps.
- Mode C (Migration): start Bedrock-only; insert ACVPS Gateway; transition to provider-agnostic enforcement without app changes.
- Mode D (Dual-layer): defense-in-depth with EthicalZen pre/post checks and Bedrock at-inference checks.
- Mode E (Governance): define policies once in SentryWorks; auto-configure both EthicalZen and Bedrock enforcement.
- Mode F (Observability bridge): unified telemetry across Bedrock/OpenAI/Groq/custom while preserving native enforcement where available.

## Problem space

- Bedrock Guardrails are strong for general content moderation and some PII, but customers often need domain-specific and workflow-specific controls (finance, healthcare, legal, academic integrity, fraud).
- Large enterprises run multiple LLM providers simultaneously. Provider-specific guardrails create lock-in, inconsistent enforcement, and fragmented audit trails.
- Teams struggle with guardrail lifecycle at scale (draft → review → deploy → version pinning → retirement), especially across dozens of apps and tenants.
- Regulated industries require demonstrable controls: deterministic policy contracts, evidence collection, and provable configuration drift detection.

## Key tenets

- Defense in depth: combine pre-inference, at-inference, and post-inference controls where possible.
- Deterministic contracts: encode non-negotiable rules as explicit, testable contracts that survive provider changes.
- Policy once, enforce everywhere: centralize policy in SentryWorks; translate to provider-specific mechanisms.
- Transparent insertion: ACVPS Gateway acts as a low-latency proxy so applications do not need refactors to adopt new safety layers.

- Observable by default: every decision should produce metrics, traces, and an audit record with version pinning.
- Multi-tenant first: guardrails, policies, certificates, and telemetry must be tenant-isolated with delegated administration.

## Core integration building blocks

- AWS Bedrock Guardrails: native runtime enforcement attached to InvokeModel via guardrailIdentifier + guardrailVersion.
- ACVPS Gateway: transparent proxy for request/response interception, routing (e.g., X-Target-Endpoint), and enforcement chaining.
- EthicalZen Smart Guardrails: domain ML classifiers, embedding-based detection, gap discovery (FMA), and deterministic contracts.
- SentryWorks Policy Hub: policy-as-code governance, audit requirements, and cross-platform policy distribution.
- Alex Agent (EthicalZen): natural language guardrail designer that compiles policies into Smart Guardrails and contracts.
- Certificates & attestation: certificate-based deployment controls and evidence of compliance state.
- Observability: unified metrics/traces (Prometheus/OTel) and guardrail effectiveness analytics across providers.

## Integration modes

Each mode is a deployable pattern. Organizations may run multiple modes simultaneously by business unit, app tier, or risk class.

### Mode A: Complement mode - EthicalZen adds what Bedrock lacks

Use when the customer is committed to Bedrock and wants additional domain and enterprise controls without changing their existing Bedrock Guardrails setup.

#### Reference flow

```
Customer App → ACVPS Gateway → [EthicalZen Guardrails] → Bedrock InvokeModel (w/ Bedrock Guardrails)
                                          ↓
                                Smart Guardrails:
                                - Domain-specific (finance, healthcare, legal)
                                - FMA-discovered gaps
                                - ML classification + embeddings
                                - Deterministic contract enforcement
```

### Customer outcome

- Bedrock's native content and basic PII filters, plus specialized Smart Guardrails not offered natively (e.g., insurance fraud detection, medical advice safety, academic integrity, bias checks, token cost limiting).
- A single enforcement hop via ACVPS without re-architecting the application.

### Role split

| Component | Primary responsibility |
|---|---|
| AWS Bedrock Guardrails | General content filters, basic PII, denied topics, word filters. |
| EthicalZen Smart Guardrails | Domain ML guardrails, FMA gap analysis, deterministic contracts, prompt leakage detection, custom embedding classifiers. |
| ACVPS Gateway | Traffic proxy, chaining, logging, and policy version pinning. |

### Example use case

Healthcare assistant uses Bedrock Guardrails for general moderation and EthicalZen Smart Guardrails for medical advice safety, mental-health crisis detection, and HIPAA-specific PII patterns that exceed Bedrock defaults.

### Implementation notes

- Deploy ACVPS as an L7 proxy (sidecar, ingress, or API gateway plugin) and route Bedrock traffic through it.
- Run EthicalZen pre-checks before InvokeModel; let Bedrock Guardrails run at inference; optionally run EthicalZen post-checks for contract validation.
- Pin contract versions and guardrail versions per environment (dev/stage/prod).

## Mode B: Orchestration mode - EthicalZen manages Bedrock Guardrails via API

Use when the customer wants a single workflow to design and lifecycle-manage Bedrock Guardrails at scale (many apps/tenants) rather than configuring each guardrail in the AWS Console.

### Reference flow

```
SentryWorks Policy Engine
        ↓ (auto-configure)
EthicalZen Portal → AWS Bedrock CreateGuardrail API
        ↓                    ↓
  DC Contracts          guardrailIdentifier
```

```
        ↓                    ↓
ACVPS Gateway → InvokeModel + guardrailVersion
```

### Customer outcome

- Single pane of glass: design guardrails in EthicalZen (Alex Agent), provision them into Bedrock via API, and manage draft→active→retired from one platform.
- Consistent version pinning across applications using guardrailIdentifier + guardrailVersion.

### Role split

| Component | Primary responsibility |
|---|---|
| EthicalZen | Guardrail design (Alex Agent + FMA), lifecycle management, multi-tenant governance, cross-provider orchestration. |
| AWS Bedrock | Runtime enforcement for Bedrock-hosted models using guardrailIdentifier + guardrailVersion. |
| SentryWorks | Optional: upstream policy definition feeding EthicalZen guardrail designs. |

### Example use case

Enterprise with 50 Bedrock-powered apps designs guardrails conversationally in EthicalZen, runs FMA to discover gaps, then auto-pushes configurations to Bedrock using CreateGuardrail/UpdateGuardrail APIs. One contract governs all apps.

### Implementation notes

- Implement a Bedrock Guardrails provisioning service in EthicalZen with tenant-aware IAM roles and least-privilege permissions.
- Store guardrail metadata (identifier, versions, last-updated, policy hash) for drift detection and rollback.
- Expose promotion workflows (draft→review→active) with approvals and automated tests.

## Mode C: Migration mode - from Bedrock-only to EthicalZen (multi-provider)

Use when the customer starts with Bedrock Guardrails but wants to reduce provider lock-in and enforce equivalent policies across multiple LLM vendors without changing application code.

### Reference flow

```
Phase 1: App → Bedrock (w/ Bedrock Guardrails)
Phase 2: App → ACVPS Gateway → Bedrock (Bedrock Guardrails + EthicalZen
monitoring)
```

```
Phase 3: App → ACVPS Gateway → [EthicalZen Guardrails] → Bedrock (no Bedrock guardrails)
Phase 4: App → ACVPS Gateway → [EthicalZen Guardrails] → ANY LLM (Bedrock/OpenAI/Groq/custom)
```

### Customer outcome

- Provider independence: the same safety posture across providers using EthicalZen contracts and Smart Guardrails.
- Incremental adoption with minimal disruption (monitoring first, then enforcement, then provider routing).

### Role split

| Component | Primary responsibility |
|---|---|
| ACVPS Gateway | Transparent insertion and provider routing via headers (e.g., X-Target-Endpoint). |
| EthicalZen | Guardrail parity mapping, deterministic contracts to preserve continuity, provider-agnostic enforcement. |
| AWS Bedrock | Gradually reduced role from primary enforcement to optional monitoring to removed. |

### Example use case

Fintech using Bedrock Guardrails adds direct provider routes (e.g., Anthropic Claude API and Groq for cost). They insert ACVPS, replicate Bedrock rules as EthicalZen Smart Guardrails and contracts, then route to any provider while keeping a consistent safety posture.

### Implementation notes

- Start with shadow mode: EthicalZen evaluates but does not block; compare results against Bedrock block/allow decisions.
- Automate parity tests: generate a policy test suite and validate equivalence before switching enforcement.
- Introduce provider routing rules by workload (latency tier, cost tier, geography, compliance tier).

## Mode D: Dual-layer mode - defense in depth (both simultaneously)

Use for regulated or high-risk workloads where multiple layers of validation are required (input validation, native at-inference checks, and output contract enforcement).

### Reference flow

```
Customer App
  ↓
ACVPS Gateway → [Layer 1: EthicalZen PRE-check]
  ↓               - Smart Guardrails (ML classification)
  ↓               - Prompt injection detection
  ↓               - Deterministic contract enforcement
  ↓
Bedrock InvokeModel → [Layer 2: Bedrock AT-inference]
  ↓                     - Content filters
  ↓                     - Contextual grounding
  ↓                     - PII redaction
  ↓
ACVPS Gateway → [Layer 3: EthicalZen POST-check]
  ↓               - Response validation
  ↓               - Contract envelope constraints
  ↓               - Certificate compliance
  ↓
Customer App (safe response)
```

### Customer outcome

- Maximum safety: three validation layers with consistent audit evidence and version pinning.
- Clear separation of duties: Bedrock provides native runtime controls; EthicalZen enforces domain and contract constraints before and after inference.

### Role split

| Component | Primary responsibility |
|---|---|
| EthicalZen | Pre-flight guardrails and post-flight validation, audit trail, certificate enforcement. |
| AWS Bedrock | At-inference moderation, grounding checks, and native PII redaction. |
| ACVPS Gateway | Chaining, latency management, and observability across layers. |

### Example use case

A regulated bank needs SOC 2 and OCC-aligned controls. EthicalZen contracts enforce financial advice rules (pre + post), while Bedrock handles native moderation and grounding. Compliance evidence includes EthicalZen certificates and Bedrock guardrail version pinning.

### Implementation notes

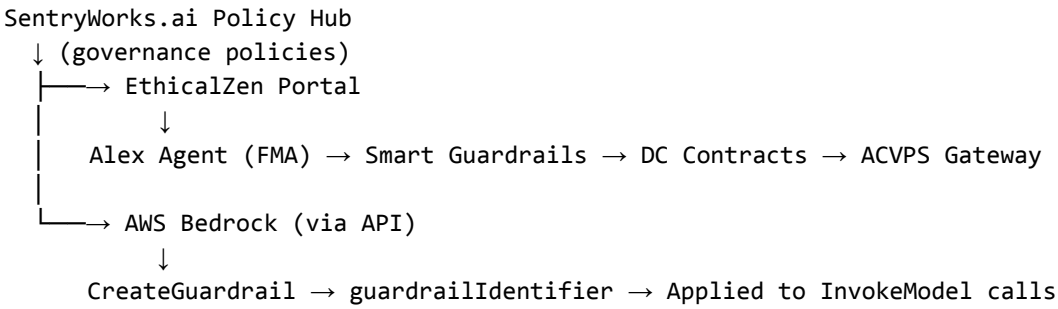
- Keep latency budgets explicit (per-layer SLOs) and allow tiered enforcement (strict for prod, relaxed for dev).

- Use structured refusal responses and standardized reason codes to reduce support burden.
- Instrument block/allow decisions at every layer for drift and incident response.

## Mode E: Governance mode - SentryWorks auto-configures both platforms

Use when an organization needs policy-as-code governance: define safety policies once and have them compiled and applied across both Bedrock-native guardrails and EthicalZen provider-agnostic guardrails.

### Reference flow




### Customer outcome

- Policy once, enforce everywhere: unified audit trail and compliance reporting across multiple providers.
- Automated translation of governance requirements into runtime controls with minimal manual configuration.

### Role split

| Component | Primary responsibility |
|---|---|
| SentryWorks | Policy definition, compliance rules, audit requirements, cross-platform governance. |
| EthicalZen | Policy-to-guardrail translation (FMA + Alex Agent), contract generation, enforcement outside Bedrock, certificate authority. |
| AWS Bedrock | Native enforcement on Bedrock-hosted models; configurations auto-managed via API. |

### Example use case

A global insurer runs 200+ AI services across Bedrock, OpenAI, and custom models. SentryWorks defines 'no unauthorized medical advice'. It auto-generates an EthicalZen Smart Guardrail for

non-Bedrock traffic and a Bedrock denied-topic/content policy for Bedrock-hosted models. One policy yields two enforcement engines.

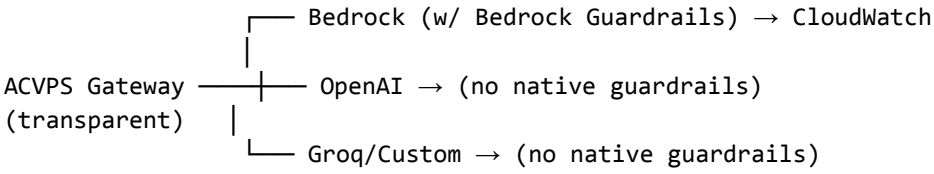
### Implementation notes

- Represent policies in a portable intermediate format (policy IR) so multiple back-ends can be compiled consistently.
- Attach evidence requirements (logs, traces, signed configs) directly to each policy object.
- Provide policy simulation and preflight testing before promotion to production.

## Mode F: Observability bridge mode - EthicalZen as unified monitoring

Use when the customer wants unified compliance monitoring across multiple LLM providers, even if enforcement remains provider-native (e.g., Bedrock Guardrails for Bedrock traffic).

### Reference flow




All traffic flows through ACVPS → Unified metrics (Prometheus/OTel) → Single dashboard

### Customer outcome

- Unified telemetry: compare Bedrock block rates vs EthicalZen risk scores; detect drift and policy effectiveness issues.
- Cross-provider reporting: one dashboard for all LLM traffic, regardless of vendor.

### Role split

| Component | Primary responsibility |
|---|---|
| EthicalZen | Telemetry collection, dashboards, effectiveness scoring, drift detection, A/B comparison of enforcement decisions. |
| AWS Bedrock | Primary runtime enforcement for Bedrock traffic. |
| ACVPS Gateway | Capture request/response pairs and emit metrics/traces in a consistent schema. |

### Example use case

An ML platform team runs five providers. Bedrock traffic uses native Bedrock Guardrails; ACVPS captures all request/response pairs for unified reporting and to identify where policies behave differently across vendors.

### Implementation notes

- Adopt OpenTelemetry spans per request with consistent attributes (tenant, app, model, policy version, decision code).
- Store aggregate metrics plus sampled payloads under strict data handling and retention policies.
- Use observability to drive continuous improvement: identify false positives/negatives and refine guardrails.

## Mode comparison matrix

Use this table to pick an adoption pattern per app portfolio, compliance tier, and desired vendor flexibility.

| Mode | Name | Best for | Primary value |
|---|---|---|---|
| A | Complement | Add domain controls while keeping Bedrock Guardrails | Wedge into Bedrock deployments with added specialized guardrails. |
| B | Orchestration | Manage/provision Bedrock Guardrails at scale from EthicalZen | Single pane of glass and lifecycle automation for Bedrock Guardrails. |
| C | Migration | Transition from Bedrock-only to provider-agnostic enforcement | Provider independence with minimal app changes. |
| D | Dual-layer | Pre + at + post inference layered controls | Defense-in-depth for regulated, high-risk workloads. |
| E | Governance | SentryWorks policy control plane across platforms | Policy-as-code governance and |

| | | | |
|---|---|---|---|
| | | | unified compliance evidence. |
| F | Observability | Unified monitoring across providers | Cross-provider analytics, drift detection, and effectiveness scoring. |

## Recommended Approach

- Start with Mode A (Complement): 'We add what Bedrock doesn't have.'
- Expand with Mode D (Dual-layer): 'Defense in depth for regulated industries.'
- Platform with Mode E (Governance): 'SentryWorks as the control plane for AI safety across providers.'

## Proposed integration modes by common use case

| Use case category | Typical constraints | Recommended modes |
|---|---|---|
| Regulated customer support (bank/insurer) | Strict refusal behavior, audit evidence, PII redaction, prompt injection resistance | D + E (plus F for analytics) |
| Healthcare guidance / triage | Medical advice safety, crisis escalation, HIPAA-style PII patterns | A → D, then E for scale |
| Enterprise internal copilots | Leakage prevention, IP/PII controls, cost limits, consistent UX | A + F, then B for lifecycle |
| Multi-provider platform team | Vendor flexibility, uniform reporting, drift detection | F + C, then E |
| High-volume experimentation teams | Fast iteration, A/B tests, preflight simulation | B + F, then E |

## Implementation roadmap (reference)

- Phase 0: Define policy IR and contract schemas; establish tenant isolation and evidence retention rules.
- Phase 1: Deploy ACVPS Gateway in shadow mode; collect baseline metrics; enable Mode F observability.
- Phase 2: Add Mode A Smart Guardrails for 2-3 high-value domains; publish a parity test suite.
- Phase 3: Enable Mode B provisioning for Bedrock Guardrails; add approval workflows and version pinning.
- Phase 4: Roll out Mode D for regulated tiers; standardize refusal UX and incident playbooks.
- Phase 5: Launch Mode E governance with SentryWorks as the control plane; expand to all providers.

## Appendix: Bedrock API touchpoints (non-exhaustive)

- CreateGuardrail / UpdateGuardrail: programmatic guardrail provisioning and updates.
- InvokeModel: apply guardrailIdentifier and guardrailVersion for runtime enforcement.
- CloudWatch metrics/logs: integrate with ACVPS telemetry for unified dashboards.