Open Source Certification Pipeline

December 2025

EQTYLAB

FINOS Fintech Open Source Foundation

Executive Summary

1. **Opportunity**: Create a program that helps organizations stand up and certify deployments in adherence to FINOS frameworks.

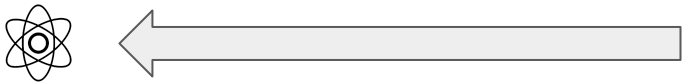
2. **Challenges**: Current guidance is often generalized and lacks structured, defined success criteria — obscuring pathways for both adoption and validation. Moreover, there is no tooling for enforcing continued compliance post-deployment.

3. **Solution**: Leverage existing FINOS work streams to create an end-to-end certification pipeline that formalizes controls, requirements to achieve compliance, and enforcement techniques.

Executive Summary

Establish FINOS's first certification pipeline by extending CALM bidirectionally.

Atomization



CALM

Cupcake

Formalize FINOS guidance by transforming text into discrete controls.

Enforce execution-dependent controls at runtime using [OPA/Rego] policy evaluation.

Desired outcome:
Each FINOS policy (e.g. AIGF) has a companion playbook with universally identifiable controls, predefined success criteria, and compliance enforcement techniques.

Goals of this Deck

1. Illustrate pipeline to **transform AIGF guidance** into discrete, machine-configurable controls.

2. Discuss how this pipeline **creates an open source certification process** that benefits FINOS and its members.

Transforming AIGF Guidance into Machine-configurable Controls

Transforming AIGF Guidance into Machine-configurable Controls

***Atomization**: Methodology for creating structured policies with discrete mandates*

1) Atomize policy

a. Extract guidance-related text from AIGF's 23 mitigations.

b. Convert relevant text into individual controls, aiming to distill guidance into binary mandates.

c. Categorize controls by type:

- Architectural (~50%)
- Runtime (~25%)
- Procedural (~25%)

***CALM**: Schema for structuring controls in a consistent, machine-configurable format*

2) Model as CALM-format controls

a. Populate control-specific attributes (e.g. ID, name, description) in CALM format.

b. Define the required evidence and enforcement method.

c. (Optional) Define references for related mitigation categories.

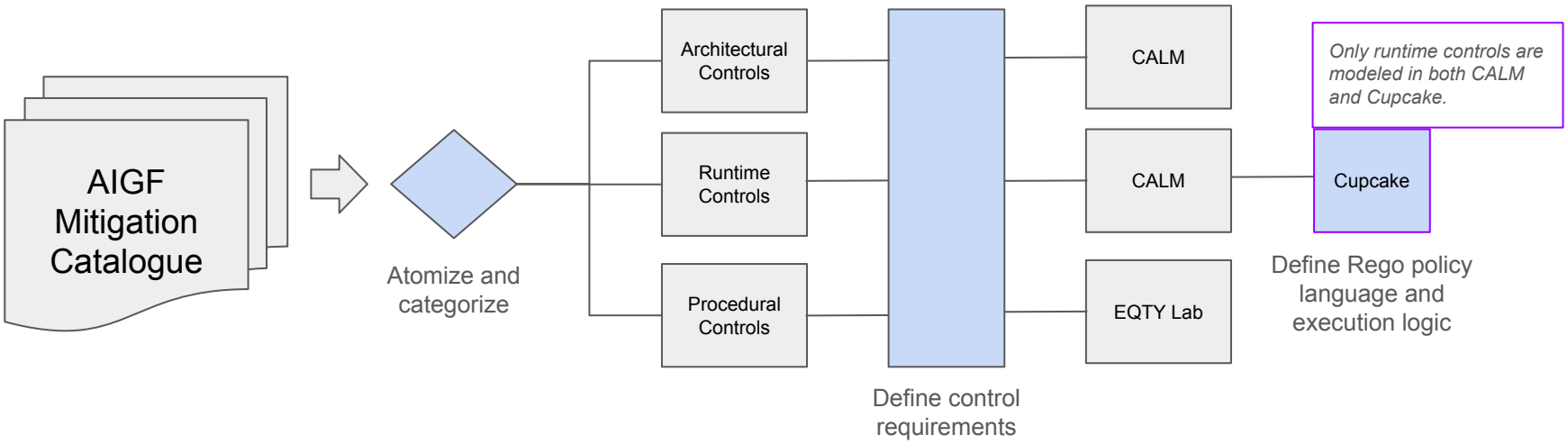
***Cupcake**: Open source enforcement layer which ensures controls are followed at runtime*

3) Model as runtime enforcement rules

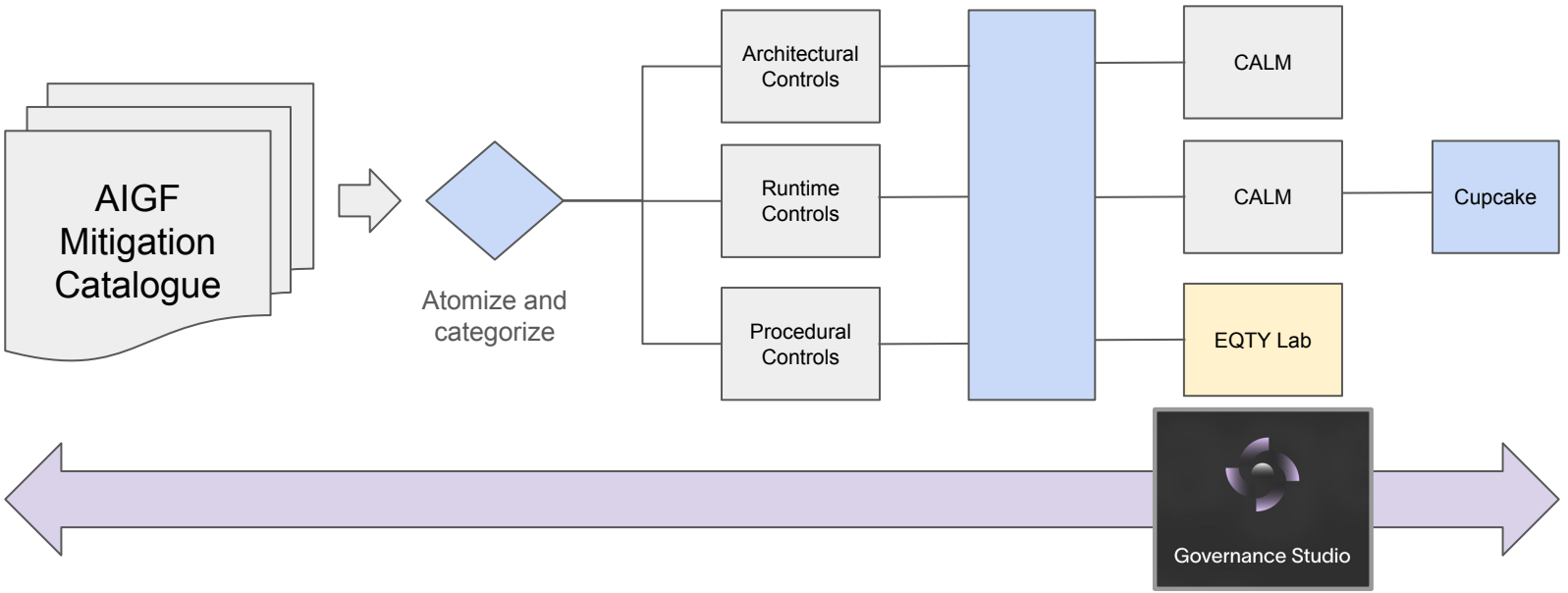
a. Translate execution-dependent controls into Rego policies that evaluate inputs, outputs, or agent actions at runtime.

b. Define the decision logic (e.g. allow, block, warn) that the enforcement layer applies based on the policy result.

Workflow Diagram



Workflow Diagram



OS certification vs. commercial certification



Open Source certification pipeline:

Certifies that **systems** are compliant against public frameworks.

Can:

1. Validate that controls are correctly defined and modeled in CALM and Cupcake.
2. Monitor that a system's policy logic behaves correctly when passed through open-source enforcement engines like Cupcake.

Cannot:

1. Produce tamper-evident proof that enforcement occurred consistently in real production environments.
2. Guarantee that runtime controls were applied on every execution without gaps.

Configuration layer

EQTYLAB

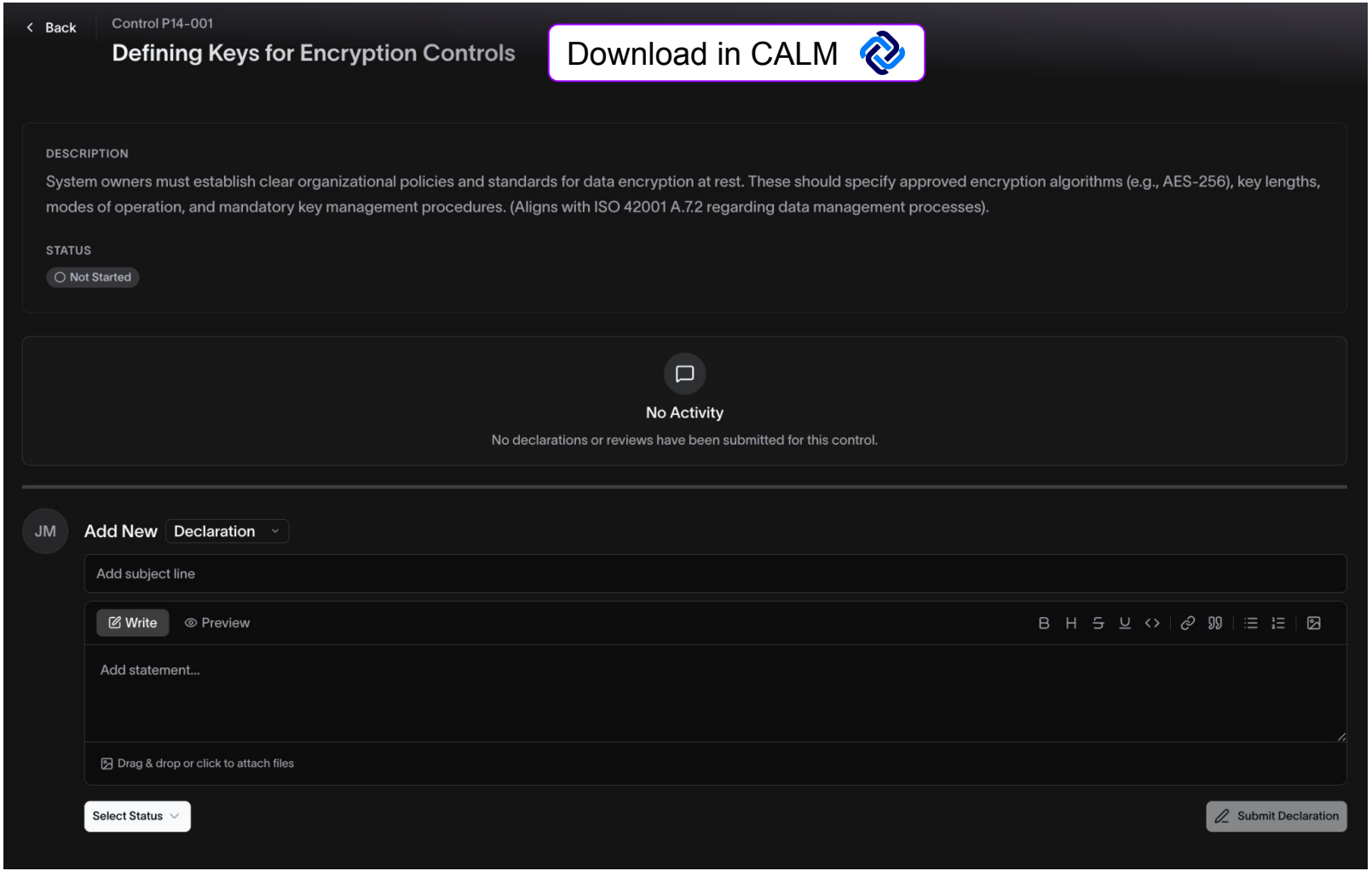
Commercial certification pipeline:

Tools that enable auditors to certify that **systems and operations** are compliant against public frameworks and **internal SOPs**.

Potential collaboration:

1. Secure evidence of compliance across data, systems, tools, and operations.
2. Create granular compliance of individual resources, not just projects.
3. Set up custom monitoring criteria for any type of control or internal process.
4. Generate audit-ready, verifiable reports instantly.
5. Collaborate in an intuitive interface that consumes both CALM and Cupcake JSONs.
6. Credentialize agents, models, components, and more.

Verification layer


Back
Control P14-001
Defining Keys for Encryption Controls
Download in CALM
DESCRIPTION
System owners must establish clear organizational policies and standards for data encryption at rest. These should specify approved encryption algorithms (e.g., AES-256), key lengths, modes of operation, and mandatory key management procedures. (Aligns with ISO 42001 A.7.2 regarding data management processes).
STATUS
Not Started
No Activity
No declarations or reviews have been submitted for this control.
JM
Add New
Declaration
Add subject line
Write
Preview
Add statement...
Drag & drop or click to attach files
Select Status
Submit Declaration

Aligning Initiatives across FINOS

Various projects across FINOS are pushing towards creating automated, machine-readable compliance workflows.

The certification pipeline leverages and compliments these existing work streams.

[Idea]: Model AI Governance Guidance in CALM (with Example) #236

New issue

Open

jpgough-ms opened on Oct 23 — Member

Contact Details

james.gough@morganstanley.com

What is the idea

Model AI Governance Guidance in CALM

Assignees

No one assigned

Labels

Idea

Type

No type

Example configuration in CALM

```
{
  "$schema": "https://schemas.company.com/controls/air-op-004-hallucination-prevention.json",
  "validation-strategy": "cross-reference",
  "confidence-threshold": 0.9,
  "citation-required": true,
  "fallback-behavior": "escalate-to-human",
  "monitoring": {
    "accuracy-tracking": true,
    "feedback-loop": true,
    "alert-threshold": 10
  }
}
```

Why is it a good idea

Outcome from NYC OSFF Hack day
Combines initiatives in FINOS

"Combines initiatives across FINOS"
Let's explore…

Gemara: GRC Engineering Model for Automated Risk Assessment

> Pronounced: Juh-MAH-ruh (think 💎)

Comprehensive model for converting written governance controls into machine-readable, automated policy rules

FINOS labs

AI Reference Architecture Library

This project consists of a collection of FSI-specific AI reference architectures. The library is part of the larger **AI Governance Framework** ecosystem and will both leverage, and be leveraged by, other framework components. Each architecture is designed and curated by a representative pool of professionals from FSIs, system integrators, and technology providers, and will be externally validated by a small group of AI domain experts.

Each architecture will be threat-modelled, with risks and mitigations taken from the AI Governance Framework catalogue, providing a practical and unified view of the applicable security baseline to design, deploy, and operate the system within agreed risk tolerances.

It is critical for the project that our output is usable and applicable to the FSI vertical. To ensure collaboration is effective, industry-standard, and unambiguous, we will establish the following guardrails:

1. Rigour in definitions
2. Agreement on naming conventions (e.g., “risk” vs “threat”)
3. Agreement on architecture layers (e.g., data, inference, agent), components within them, and data flows

Taxonomy for specific use cases

FINOS Fintech Open Source Foundation

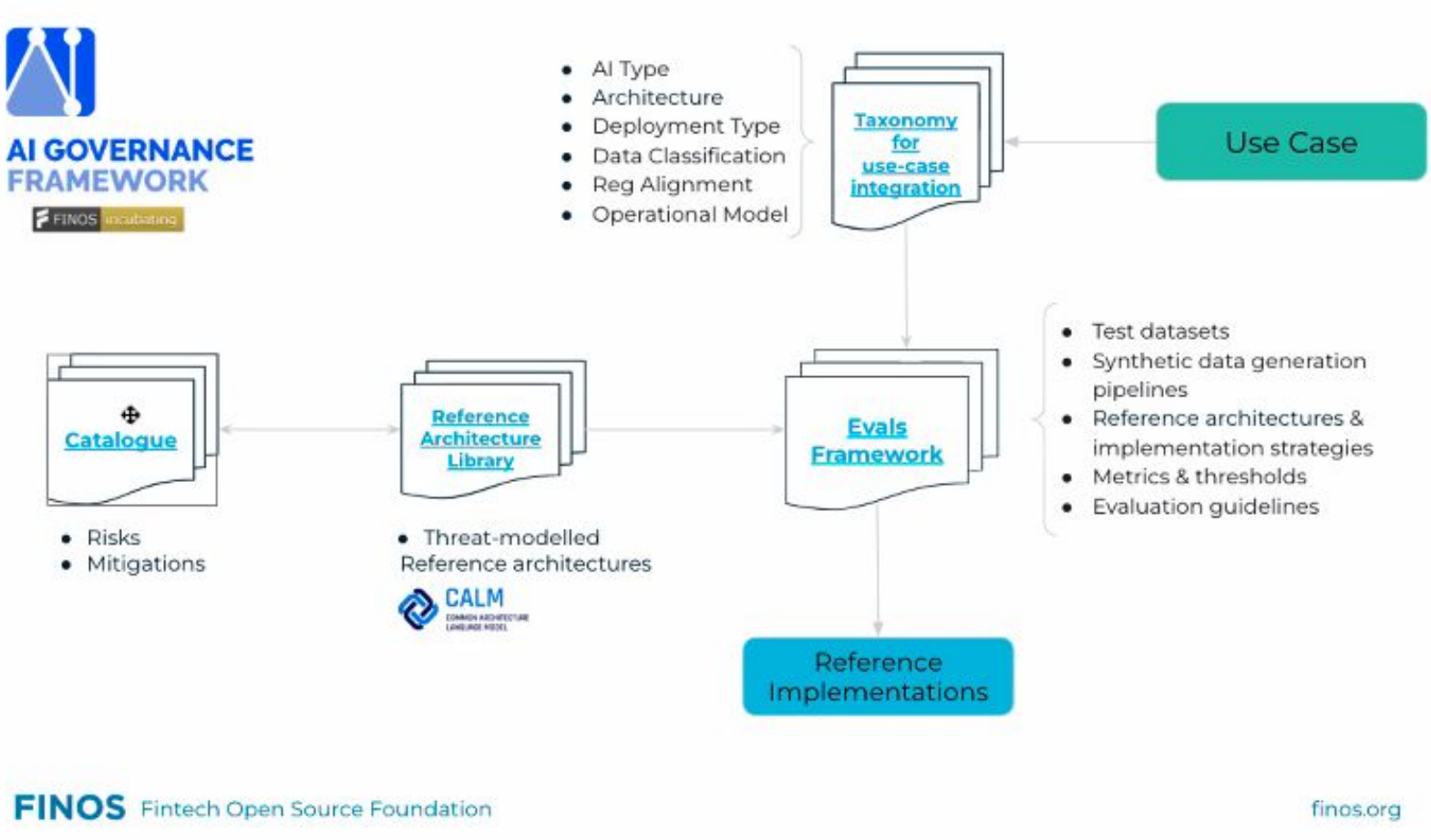
FINOS AI Strategic Initiative

Building a Taxonomy for AI Evals in Finance

From use cases to benchmarks: a shared framework

Defining evals for specific use cases

THE LINUX FOUNDATION


AI GOVERNANCE FRAMEWORK
FINOS incubating
AI Type
Architecture
Deployment Type
Data Classification
Reg Alignment
Operational Model
Taxonomy for use-case integration
Use Case
Catalogue
Risks
Mitigations
Reference Architecture Library
Threat-modelled Reference architectures
CALM
Evals Framework
Test datasets
Synthetic data generation pipelines
Reference architectures & implementation strategies
Metrics & thresholds
Evaluation guidelines
Reference Implementations
FINOS Fintech Open Source Foundation
finos.org

Example Control Workflow

Example Control Workflow

In this section, we'll explore each transformation step in greater detail using real AIGF guidance text.

All controls — architectural, procedural, and runtime — are defined in CALM, but only runtime controls are enforced through Cupcake.

Architectural Control: CALM only

Step 1: Atomize

Atomization goal: create a consistent template for each framework.

For each mandate, define:

1. Responsible party / type (e.g. System owners, System owners using RAG)
2. Obligation type (must or may)
3. Binary or near-binary mandate

AIR-PREV-014

Encryption of AI Data at Rest

Implementation Guidance

Effective implementation of data at rest encryption for AI systems involves the following:

1. Define Policies and Standards

- Establish clear organizational policies and standards for data encryption at rest. These should specify approved encryption algorithms (e.g., AES-256), key lengths, modes of operation, and mandatory key management procedures. (Aligns with ISO 42001 A.7.2 regarding data management processes).

System owners must establish clear organizational policies and standards for data encryption at rest. These should specify approved encryption algorithms (e.g., AES-256), key lengths, modes of operation, and mandatory key management procedures.

Step 2: Model in CALM

The atomization methodology already maps to the CALM format.

P14-001 ⇨ Control ID

Establishing Standards for Encrypting Data at Rest ⇨ Control title

“System owners…” ⇨ Control description

Binary, control-specific ⇨ Control requirements

Full example CALM-formatted control:

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "http://calm.finos.org/controls/aigf/security/schema/encryption-at-rest.json",
  "title": "Encryption at Rest Standard",
  "type": "object",
  "allOf": [
    {
      "$ref": "http://calm.finos.org/controls/2025-03/meta/control-requirement.json"
    }
  ],
  "properties": {
    "control-id": {
      "const": "P14-001"
    },
    "name": {
      "const": "Establishing Standards for Encrypting Data at Rest"
    },
    "description": {
      "const": "System owners must establish clear organizational policies and standards for data encryption at rest. These should specify approved encryption algorithms (e.g., AES-256), key lengths, modes of operation, and mandatory key management procedures."
    },
    "approvedAlgorithms": {
      "type": "array",
      "items": {
        "enum": ["AES-256", "AES-192", "AES-128-GCM"]
      },
      "minItems": 1
    },
    "minimumKeyLength": {
      "type": "integer",
      "enum": [128, 192, 256]
    },
    "modeOfOperation": {
      "enum": ["GCM", "CBC", "CTR"]
    },
    "keyManagementSystem": {
      "type": "string",
      "description": "Approved KMS solution (e.g., AWS KMS, Azure Key Vault, HashiCorp Vault)"
    },
    "keyRotationPolicy": {
      "type": "string",
      "enum": ["90-days", "180-days", "365-days"]
    }
  },
  "required": [
    "control-id",
    "name",
    "description",
    "approvedAlgorithms",
    "minimumKeyLength",
    "modeOfOperation",
    "keyManagementSystem",
    "keyRotationPolicy"
  ]
}
```

Step 2: Model in CALM

CALM modeling goal: conceive of standardized evidence that can be used to satisfy the atomized control.

In our example control, we can help define the following for FSIs to automatically implement or validate a compliant deployment architecture:

Control-requirements:

approvedAlgorithms: List of permitted encryption algorithms (e.g., AES-256, AES-192, AES-128-GCM)

minimumKeyLength: Minimum acceptable key length in bits (128, 192, or 256)

modeOfOperation: Approved cipher modes (GCM, CBC, CTR)

keyManagementSystem: Designated key management solution (e.g., AWS KMS, Azure Key Vault, HashiCorp Vault)

keyRotationPolicy: Frequency of key rotation (90-days, 180-days, 365-days)

System owners must establish clear organizational policies and standards for data encryption at rest. These should specify approved encryption algorithms (e.g., AES-256), key lengths, modes of operation, and mandatory key management procedures.

Runtime Control: CALM + Cupcake

Step 1: Atomize

Atomization goal: create a consistent template for each framework.

For each mandate, define:

1. Responsible party / type (e.g. System owners using RAG)
2. Obligation type (must or may)
3. Binary or near-binary mandate

AIR-PREV-017

AI Firewall Implementation and Management

The primary purpose of an AI Firewall is to mitigate these emerging AI-specific threats, including but not limited to:

- **Malicious Inputs:** Such as Prompt Injection attacks intended to manipulate model behavior or execute unauthorized actions.
- **Data Exfiltration and Leakage:** Preventing sensitive information (e.g., PII, confidential corporate data) from being inadvertently or maliciously extracted through model inputs or outputs

CALM can define the requirement, but it cannot enforce it post-deployment.

System owners must prevent sensitive information (e.g., PII, confidential corporate data) from being inadvertently or maliciously extracted through model inputs or outputs.

Step 2: Model in CALM

CALM modeling goal: conceive of standardized evidence that can be used to satisfy the atomized control.

In our example control, we can help define the following for FSIs to automatically implement or validate compliant runtime behavior:

Control-requirements:
dataTypesToProtect:
List of sensitive data categories that must be detected and protected
(e.g., PII, ConfidentialCorporate, FinancialData)

inputScanningEnabled:
Indicates whether inbound prompts must be scanned for sensitive information
(e.g., true)

outputScanningEnabled:
Indicates whether model outputs must be scanned for sensitive information
(e.g., true)

minimumDetectionAccuracy:
Minimum acceptable accuracy threshold for sensitive-data detection tooling
(e.g., 0.90)

enforcementAction:
Required action when sensitive information or extraction attempts are detected
(e.g., block)

allowedRolesForSensitiveOutput:
Roles permitted to receive sensitive information in outputs
(e.g., [] for none)

loggingRequired:
Indicates whether all violations must be logged
(e.g., true)

auditLogFields:
Required fields that must appear in violation logs
(e.g., timestamp, userId, detectedType, action)

Example CALM-formatted control:

```
{
  "control-id": "AIR-OP-003",
  "name": "Prevent Sensitive Information Extraction",
  "description": "System owners must prevent sensitive information from being inadvertently or maliciously extracted through model inputs or outputs.",

  "dataTypesToProtect": ["PII", "ConfidentialCorporate", "FinancialData"],
  "inputScanningEnabled": true,
  "outputScanningEnabled": true,
  "minimumDetectionAccuracy": 0.90,
  "enforcementAction": "block",
  "allowedRolesForSensitiveOutput": [],
  "loggingRequired": true,
  "auditLogFields": ["timestamp", "userId", "detectedType", "action"]
}
```

Step 3: Model in Cupcake

Cupcake modeling goal: Define the detection logic and enforcement actions required to prevent violations at runtime.

In our example control, we can help define the following for FSIs to automatically monitor and block sensitive data leaks without human intervention:

Detection Inputs

- Input scans provide Rego with extraction-attempt signals.
- Output scans provide detected data types and confidence scores.

Policy Evaluation (Rego)

- Check whether detected types match dataTypesToProtect.
- Verify detection accuracy meets the defined threshold.
- Confirm whether the requesting user is authorized.
- Return allow/deny/warn based on policy logic.

Enforcement (Cupcake)

- Block, warn, or allow based on Rego's decision.
- Log violation details according to the control requirements.

```
package controls.prevent_sensitive_extraction

cfg := input.signal.prevent_sensitive_extraction

deny[msg] {
    cfg.outputScanningEnabled
    cfg.enforcementAction == "block"

    some dt
    dt := input.signal.output.detectedTypes[_]
    dt == cfg.dataTypesToProtect[_]

    input.signal.output.detectionAccuracy >= cfg.minimumDetectionAccuracy

    not user_is_allowed

    msg := "Blocked: sensitive information detected in model output."
}

deny[msg] {
    cfg.inputScanningEnabled
    cfg.enforcementAction == "block"

    input.signal.input.extractionAttempt == true
    not user_is_allowed

    msg := "Blocked: unauthorized extraction attempt."
}

user_is_allowed {
    some r
    r := cfg.allowedRolesForSensitiveOutput[_]
    r == input.user.role
}
```

Category	Control	Rationale
Architectural	System owners must implement separate, isolated AI systems or environments for datasets or knowledge sources containing exceptionally sensitive information that cannot be adequately protected through standard cleansing or anonymization techniques.	This is about how systems and environments are **physically/logically separated**. It's an infra / topology decision, not something enforced per-inference.
Architectural	System owners must segment data and AI system access based on clearly defined access domains that mirror the organization's existing data classification and access control structures.	This defines **segmentation boundaries and access domains** in the architecture. Once configured (IAM, networks, stores), it doesn't require runtime content inspection.
Architectural	System owners must create distinct AI models and associated data stores (e.g., separate vector databases for RAG systems) with much stricter access controls, enhanced encryption, and limited network connectivity for datasets or knowledge sources containing exceptionally sensitive information that cannot be adequately protected through standard cleansing or anonymization techniques.	This mandates **separate models and data stores with stronger controls**. It's about how you deploy and wire systems, not how each response is filtered.

Category	Control	Rationale
Runtime	System owners must ensure that responses and information generated by the AI system are monitored and filtered before being presented to users or integrated into other systems as an additional layer of defense.	This explicitly requires **monitoring and filtering of outputs** before delivery. That implies real-time inspection and enforcement — a Cupcake-style runtime guardrail.
Runtime	System owners must detect and remove any sensitive data that might have inadvertently bypassed the initial input cleansing stages or was unexpectedly reconstructed or inferred by the AI model during its processing.	This is about **catching and stripping sensitive content at inference time**, including reconstructed data. That's dynamic behavior, not static configuration.
Runtime	System owners must implement intelligent filtering that considers the context of the user's query and their authorization level to determine what information should be included in the response.	"Intelligent filtering" based on **live context + user auth** is exactly a runtime policy. It needs per-request logic, not just upfront architecture.

EQTYLAB

Category	Control (exact text)	Rationale
Procedural	System owners must periodically audit the effectiveness of data filtering processes by sampling processed data and checking for any sensitive information that may have been missed.	This is a **recurring audit activity** done by people or batch processes, not an always-on guardrail. It checks that controls work; it doesn't enforce them in real time.
Procedural	System owners must establish mechanisms for users and reviewers to report instances where sensitive information may have been inappropriately exposed, using this feedback to improve filtering algorithms and processes.	This sets up a **feedback / escalation process** for incidents. It's governance and continuous improvement, not runtime enforcement.
Procedural	System owners must conduct periodic audits of data classification accuracy across key repositories and assess the effectiveness of data quality management processes for AI data sources.	Again, this is **periodic review and assessment**, focused on data governance quality. It sits in process and audit land, not in the runtime path.

Benefits for FINOS

What are the benefits for FINOS?

1. Accelerate AIGF Adoption

a. **Reduce implementation friction**: FSIs can deploy controls immediately rather than spending months interpreting policy text.

b. **Create reference implementations**: Demonstrate AIGF compliance in production environments.

2. Enhance Certification Accuracy & Consistency

a. **Make AIGF audit-friendly**: Give auditors, vendors, and FSIs a shared structure for reviewing evidence and compliance, reducing interpretation gaps and accelerating certification reviews.

b. **Reduce certification ambiguity**: Replace policy language with structured CALM schemas that make control intent, scope, and expected evidence unambiguous.

c. **Shift from point-in-time to continuous compliance**: Replace point-in-time audit snapshots with ongoing control validation, giving GRC teams and auditors real-time visibility into compliance status.

3. Amplify FSI Collaboration

a. **Build a shared control library**: FSIs contribute and refine CALM-formatted controls specific to financial use cases (credit underwriting, fraud detection, trading algorithms), creating a community-maintained repository that accelerates implementation for all members while reducing redundant policy development work.

b. **Coordinate vendor requirements**: Unified standards give FSIs collective bargaining power to demand CALM-compatible controls from AI providers and cloud vendors.

Benefits for FINOS Members

What are the benefits for FINOS Members?

1. Accelerate ROI for AI and Agents

a. **Get to market faster:** Deploy AI systems immediately with pre-built, certifiable controls instead of delayed custom policy interpretation.

b. **Out-of-the-box trust:** Deploy AI systems immediately with pre-built, certifiable controls that match industry standards and earn trust with partners and customers alike.

2. Strengthen Security Posture

a. **Block high-risk agent actions**: Apply predefined guardrails that prevent misuse, unsafe tool access, and unauthorized data movement.

b. **Mitigate model manipulation**: Use FINOS-aligned security controls to detect prompt injection, jailbreak attempts, and drift in model behavior.

c. **Accelerate incident response**: Gain traceability across model decisions, agent actions, and configuration changes to shorten investigation cycles.

3. Streamline Deployment & Continuous Compliance

a. **Guarantee configuration fidelity**: Ensure that every production system is running the exact approved model version and configuration, eliminating version drift and unauthorized changes.

b. **Enable continuous auditing:** Shift from costly, intermittent audit cycles to real-time, continuous compliance monitoring with full visibility into risk status.

Open Questions

Open Questions

1. What is an active use case for CALM?
2. Per the CALM v01 announcement below, what software is being used to manage, track, and validate these (if any)?
    a. “CALM is already in use in several firms, where it is being used to document architectures of existing systems, enable pattern based automated security approvals and has already underpinned well in excess of 2000 application deployments.”
3. Who will be responsible for defining and approving the CALM control-requirements and Rego policy language?
4. How could this pipeline interact with Gemara?
5. How could FINOS independently validate a deployment adheres to the required architecture?
6. How might this work fold into other certification types FINOS is interested in providing? There was mention of certifying operators as well.

Appendix:
Commercial Solutions with EQTY Lab

Refresh: About EQTY

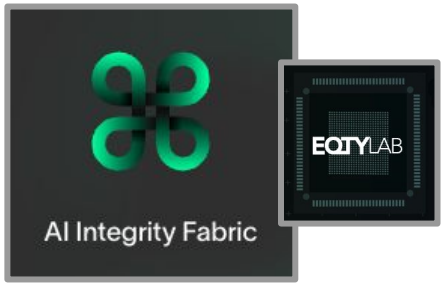
User-facing GRC tools

Lineage Explorer is an interactive interface for visualizing and validating your AI's lifecycle.

Governance Studio is a unified platform for automating and proving compliance with policies.

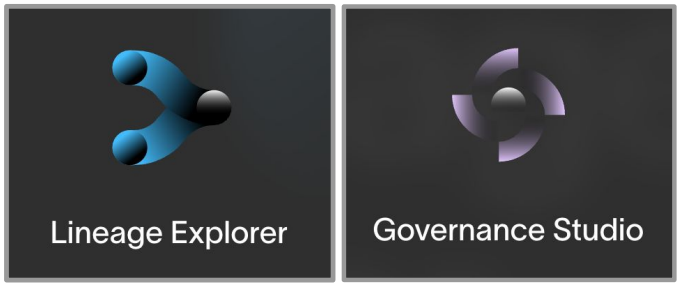
Powered by integrity

AI Integrity Fabric underwrites integrity across your entire workflow through advanced cryptography.



Verifiable Compute creates hardware-level attestations to validate and notarize inputs and outputs at runtime.

Refresh: About EQTY



Value Prop

Conventional GRC = inefficient and doesn't scale to AI	EQTY Integrity Suite = purpose-built for AI and agents
❌ Screenshots as evidence = tamperable and untrustworthy	✅ Code-based logs that are tamperproof and verifiable
❌ Manual checks = inaccurate and error prone	✅ Automated continuous monitoring against any control
❌ Endless email chains = time cost + impossible to track progress	✅ Unified, collaborative compliance dashboard

Refresh: About EQTY

Value Prop

Conventional supply chain infra = attack-prone and unverifiable	EQTY Integrity Fabric + VCOMP = secure and trusted executions
❌ Model versions and configs logged in plain text = mutable	✅ Every model build, dataset, and inference hashed and signed as integrity statements
❌ Runtime environments trusted “as is” = hidden risk of tampering or misconfig	✅ Hardware-backed attestations prove code ran in a secure enclave
❌ Provenance scattered across siloed systems = fragmented audit trail	✅ Unified integrity manifests link data, models, and runtime into one verifiable chain

Procedural Control: CALM + Governance Studio

Example Control: Data Filtering

AIR-PREV-002

Atomized control: *System owners must periodically audit the effectiveness of data filtering processes by sampling processed data and checking for any sensitive information that may have been missed.*

1. What evidence is being requested?	2. Why is unverifiable evidence insufficient?	3. How does integrity-backed evidence strengthen control artifacts?
This control requires leadership to demonstrate that data filtering processes function as expected over time by examining processed data samples to confirm that no sensitive data remains. **Our example evidence**: Management must prove that a **processed data sample contains no PII** and that the dataset passed through the filtering pipeline without introducing or leaking sensitive information.	Unverifiable evidence such as **screenshots, exported samples, or human attestations** only demonstrates what a single snapshot looked like. It does not prove that the sample was authentic, representative, or untampered, nor that filtering controls worked consistently between audit periods.	**Verifiable Evidence**: Continuous checks evaluate every pipeline run for PII leakage, producing tamper-evident records of filtering performance. Hardware-signed statements confirm that processed data originated from the expected dataset and passed through the required filtering steps. Auditors can independently verify that filtering controls were applied correctly and trace data lineage throughout the entire pipeline.

Step 1: Atomize

Atomization goal: create a consistent template for each framework.

For each mandate, define:

1. Responsible party / type (e.g. System owners using RAG)
2. Obligation type (must or may)
3. Binary or near-binary mandate

AIR-PREV-002

Data Filtering From External Knowledge Bases

CALM can define the requirement, but it cannot enforce it post-deployment.

5. Monitoring and Continuous Improvement

- **Regular Review of Filtering Effectiveness:** Periodically audit the effectiveness of data filtering processes by sampling processed data and checking for any sensitive information that may have been missed.

System owners must periodically audit the effectiveness of data filtering processes by sampling processed data and checking for any sensitive information that may have been missed.

Step 2: Model in CALM

CALM modeling goal: conceive of standardized evidence that can be used to satisfy the atomized control.

In our example control, we can help define the following for FSIs to automatically implement or validate continuous compliance with data filtering requirements.

Control-requirements:

samplingFrequency:
How often processed data must be evaluated for leakage
(e.g., per-run, daily, weekly, monthly)

sensitiveDataTypes:
List of sensitive categories that must be detected and validated as absent
(e.g., PII, PHI, FinancialData)

evidenceRequired:
Types of evidence needed to demonstrate filtering effectiveness
(e.g., filteredSampleDataset, scanResults, lineageRecords, integrityStatements)

leakageThreshold:
Maximum allowable count of detected sensitive items in the processed data sample
(e.g., 0)

Example CALM-formatted control:

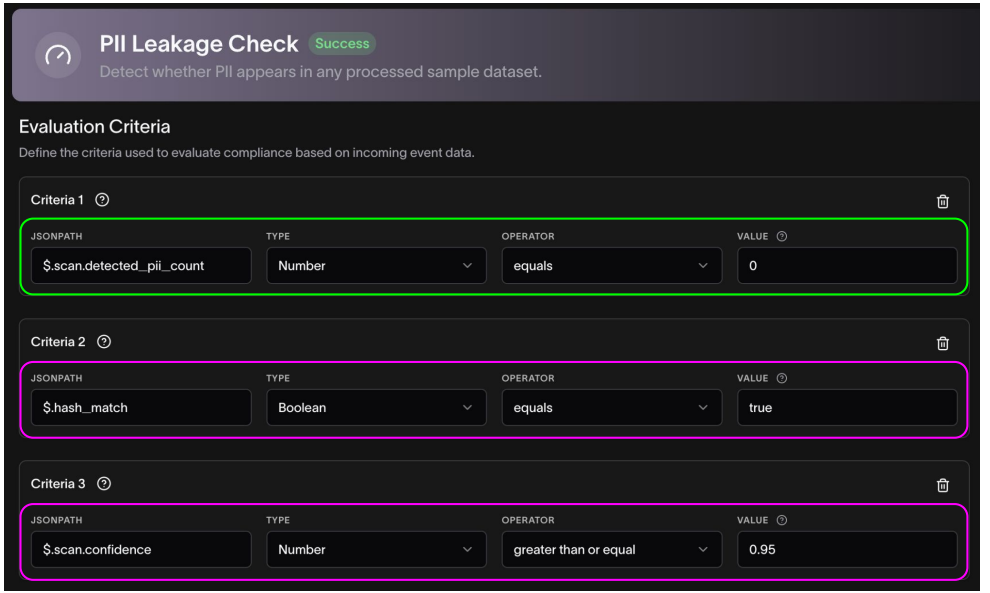
```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "control-id": "AIR-PREV-002",
  "name": "Periodic Verification of Data Filtering Effectiveness",
  "description": "System owners must periodically audit the effectiveness of data filtering processes by sampling processed
data and checking for any sensitive information that may have been missed.",
  "control-requirements": {
    "samplingFrequency": {
      "type": "string",
      "enum": ["per-run", "daily", "weekly", "monthly"]
    },
    "sensitiveDataTypes": {
      "type": "array",
      "items": { "enum": ["PII", "PHI", "FinancialData"] },
      "minItems": 1
    },
    "evidenceRequired": {
      "type": "array",
      "items": {
        "enum": [
          "filteredSampleDataset",
          "scanResults",
          "lineageRecords",
          "integrityStatements"
        ]
      }
    },
    "leakageThreshold": {
      "type": "integer",
      "description": "Maximum allowed count of detected sensitive items in the post-filter sample.",
      "default": 0
    }
  },
  "required": [
    "control-id",
    "name",
    "description",
    "control-requirements"
  ]
}
```

Step 3: Upload to Governance Studio

Upload to Governance Studio goal: auto-populate Governance Studio indicators with CALM control-requirements to continuously monitor data leakage.

Example CALM-formatted control:

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "control-id": "AIR-PREV-002",
  "name": "Periodic Verification of Data Filtering Effectiveness",
  "description": "System owners must periodically audit the effectiveness of data filtering processes by sampling processed data and
checking for any sensitive information that may have been missed.",
  "control-requirements": {
    "samplingFrequency": {
      "type": "string",
      "enum": ["per-run", "daily", "weekly", "monthly"]
    },
    "sensitiveDataTypes": {
      "type": "array",
      "items": { "enum": ["PII", "PHI", "FinancialData"] },
      "minItems": 1
    },
    "evidenceRequired": {
      "type": "array",
      "items": {
        "enum": [
          "filteredSampleDataset",
          "scanResults",
          "lineageRecords",
          "integrityStatements"
        ]
      }
    },
    "leakageThreshold": {
      "type": "integer",
      "description": "Maximum allowed count of detected sensitive items in the post-filter sample.",
      "default": 0
    }
  },
  "required": [
    "control-id",
    "name",
    "description",
    "control-requirements"
  ]
}
```



Step 4: Monitor in Governance Studio

Governance Studio monitoring goal: Continuously monitor data leakage and share alerts when controls fall out of compliance.

This way, we can help FSIs to continuously audit their data filtering processes against the specific, relevant AIGF controls.