CASE STUDY CONTRIBUTION

No valid grant, no refund

A reproducible Fidacy reference deployment showing how an AI agent can propose a consequential action while a model-independent authority boundary decides whether it may execute.

DOCUMENT

Case study contribution

CONTRIBUTOR

Fidacy, operated by ZEEPCODE GROUP LLC
Lucas de Lima, Founder

FRAMEWORK

IMDA Model AI Governance Framework for Agentic AI
Version 1.5 · 20 May 2026 · updated 5 June 2026

1 · PROPOSED TITLE OF CASE STUDY

No valid grant, no refund: a cryptographically enforceable authority boundary for an AI agent

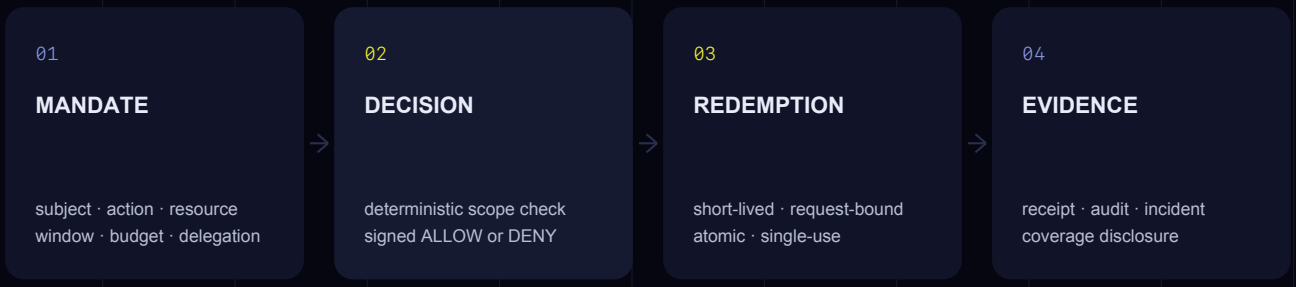
The agent may reason, recommend and request. It cannot create the authority required to release a refund. A connected executor proceeds only after the exact request receives and redeems a short-lived, single-use signed grant.

This case addresses a practical gap between governing an agent's behaviour and governing the real-world action path available to it. The control sits outside the model, prompt and memory. It is therefore evaluated by what the protected executor can release, not by what the model says it intends to do.

SAFETY PROPERTY	For a Fidacy-connected path, no valid and unredeemed request-bound grant means no release to the consequential side effect.

Case status

Fidacy-operated technical reference deployment. Component controls were executed against versioned repository code. A complete authority cycle then ran through the production API and Stripe test mode using an isolated Enterprise technical tenant. No external customer deployment or movement of live customer funds is claimed.



Best demonstrated MGF dimensions

- Dimension 1: bound action-space, resources, time, budget and delegation upfront.
- Dimension 2: keep authority with an accountable organisation, not the agent.
- Dimension 3: enforce deterministic controls outside the model and preserve runtime evidence.

2 · DESCRIPTION OF ORGANISATION

Fidacy provides authority and independently verifiable evidence infrastructure for AI agents.

Fidacy is operated by ZEEPCODE GROUP LLC. It provides a neutral control layer between AI agents and consequential actions in enterprise systems. The service is non-custodial: it does not hold customer funds or execute payment settlement. It evaluates authority, signs decisions and release grants, records evidence and exposes public verification interfaces.

How AI agents are used

Fidacy uses AI agents in engineering and operational workflows and operates reference integrations in which an agent proposes actions through MCP, SDK or API-connected tools. In this case, a support agent constructs a refund request against a named payment resource. The agent can interpret the customer request, but it cannot mint, alter or extend the organisational authority required to release the refund.

The authority service runs independently of the agent's model, prompt and memory. This separation is deliberate. The agent may be non-deterministic, misinterpret a request or be influenced by malicious content. The action boundary evaluates whether the exact proposed action is covered by active authority.

Production interfaces used by the pattern

Component	Role in the case
POST /v1/action-mandates	Creates bounded authority with subject, actions, resources, validity, budget and delegation depth.
POST /v1/action-mandates/:id/decisions	Evaluates the exact request and returns signed ALLOW or DENY evidence.
POST /v1/action-grants/redeem	Atomically accepts an eligible grant once before the connected executor proceeds.
Incident Pack	Groups mandate, decision, receipt, anchor state and explicitly disclosed correlations.
Control Coverage	Separates current connector evidence from historical hard-gate evidence and known gaps.
/.well-known/jwks.json	Allows external signature verification without an unrestricted tenant credential.

NEUTRAL ROLE	The agent proposes. The organisation grants authority. Fidacy evaluates and signs. The connected executor enforces. The downstream system proves completion.

3 · DESCRIPTION OF CASE STUDY

A refund request crosses a model-independent authority boundary before the action path can continue.

A refund is consequential because it moves value and can create a binding representation to a customer. Prompt instructions alone were treated as insufficient because the agent also has access to a tool that can initiate the action.

Reference transaction

A support-refund-agent requests stripe.refund.create against stripe:payment_intent:pi_test_123. Locally retained context is represented by a SHA-256 digest. The organisation has granted at most one matching action within a defined time window. The agent does not hold the mandate-signing key.

Bounded mandate

The accountable organisation creates a machine-readable mandate naming the agent subject, permitted action, specific resource, validity window, maximum action count and delegation limit. A child mandate may narrow but cannot widen those limits.

Decision and grant

Fidacy evaluates action, resource, context digest, mandate status, time window, budget and delegation constraints. An in-scope request receives a signed ALLOW receipt and a short-lived grant bound to the exact request. An out-of-scope, expired, revoked or exhausted request receives signed DENY evidence and no grant.

Runtime enforcement

The connected executor verifies issuer, signature, validity and exact request binding, then redeems the grant atomically immediately before the side-effect path. A repeated redemption is rejected. A mismatch between the live request and signed grant prevents release.

Evidence after the decision

The decision receipt proves issuer attribution and integrity. Redemption proves that a connected executor accepted the grant before continuing. Neither object alone proves that a payment service completed the refund. That stronger claim requires the downstream system's own operational receipt.

3.1 · EXACT AUTHORITY OBJECTS

The control evaluates structured authority, not natural-language intent.

Synthetic test vector. Identifiers are non-customer values and the context remains local.

```
{
  "subject": "support-refund-agent",
  "allow": {
    "actions": ["stripe.refund.create"],
    "resources": ["stripe:payment_intent:pi_test_123"],
    "maxActions": 1, "maxDelegationDepth": 1
  },
  "window": { "notBefore": "2026-01-01T00:00:00Z",
              "notAfter": "2026-12-31T23:59:59Z" }
}
```

Exact proposed action

```
{
  "action": "stripe.refund.create",
  "resource": "stripe:payment_intent:pi_test_123",
  "contextHash": "aaaaaaaaaaaaaaaa...aaaaaaaa"
}
```

Decision semantics

Outcome	Release object	Executor consequence
ALLOW	Signed receipt plus request-bound, expiring grant.	May proceed only after successful atomic redemption.
DENY	Signed receipt plus violated rule. No grant.	Must stop. There is no executable release object.
REPLAY	No new authority. grant_replayed.	Must stop. Side effect is not invoked again.

DATA MINIMISATION	The transcript or full business context may remain inside the deployer's infrastructure. Fidacy can bind the decision to a local SHA-256 digest, while the deployer preserves the original evidence needed to interpret that digest.

3.2 · OBSERVED ASSURANCE RUN

The complete production authority cycle passed, from attempted bypass to downstream test execution and evidence export.

The assurance run was executed on 29 August 2026. Six component tests passed, followed by nine production acceptance checks through an isolated Enterprise technical tenant. Production health, readiness and JWKS interfaces returned HTTP 200. No customer data or live funds were used.

Observed control	Method	Result
Cryptographic and policy controls	Six component tests for binding, signatures, budget, delegation and replay.	PASS · 6/6
Out-of-scope computer action	Attempt export action absent from the mandate through production API.	PASS · DENY with signed evidence
Tampered refund	Change authorised USD 24 refund to USD 240 while presenting original grant.	PASS · context_mismatch; Stripe not called
Exact authorised refund	Redeem exact grant and call the downstream executor.	PASS · EXECUTED in Stripe test mode
Durable replay refusal	Present the redeemed grant a second time.	PASS · grant_replayed; no second refund
Incident Pack	Export evidence for the executed ALLOW decision.	PASS · signed receipt and SHA-256 digest
Tenant closure	Close tenant, revoke keys and retry former credential.	PASS · production API failed closed

Evidence seal

Machine-readable file: Fidacy-IMDA-Assurance-Run-2026-08-29.json

SHA-256 df2994dbba9f4d277fe272b05ae893918b8b374725c7d681692df5d3f2dad46a

PRODUCTION RUN	Run 30cb4734 · authority f7cc9429... · isolated tenant closed after verification

Downstream evidence

The exact USD 24 refund executed against a succeeded Stripe test-mode payment. The altered USD 240 request and the replayed grant were refused before another Stripe refund call. The Incident Pack preserved the signed ALLOW receipt and evidence digest 487c78f8...9433831.

3.3 · ADVERSARIAL CONTROL MATRIX

The case is evaluated by attempted bypass, not only by the expected path.

Threat or failure	Boundary response	Assurance status
Prompt injection asks for an unapproved action	Deterministic action allowlist check returns DENY and no grant.	Component tested
Correct action targets the wrong resource	Exact resource binding prevents release.	Component tested
Request context changes after approval	contextHash mismatch invalidates the grant.	Component tested
Grant is replayed by another executor	Atomic redemption returns grant_replayed before effect invocation.	Component tested
Child agent expands its authority	Delegation validation rejects wider action, budget or depth.	Component tested
Authority is exhausted or tenant is closed	No new executable release remains available; tenant keys fail closed.	Budget component tested; tenant closure production tested
Tool bypasses every connected executor	No false enforcement claim. Coverage is NOT_OBSERVABLE.	Declared residual risk
PSP fails after authority release	Authority evidence is preserved; completion requires PSP receipt.	Declared evidence boundary

DESIGN PRINCIPLE	A control is not considered effective merely because the model was instructed to obey it. The protected side-effect path must require the release object.

3.4 · RELEVANT MGF DIMENSIONS

How the deployment converts governance recommendations into executable controls

MGF dimension	Practice demonstrated	Evidence available
1 · Assess and bound risks upfront	Action-space is bounded through action and resource allowlists, validity, budget and constrained delegation.	Versioned mandate and deterministic test vectors.
2 · Make humans meaningfully accountable	An accountable organisation issues authority. The agent cannot self-authorise an exception.	Subject attribution, mandate state and signed decision.
3 · Implement technical controls	A model-independent service evaluates the request. A connected executor requires a redeemed grant.	Signed receipt, request-bound JWS and replay refusal.
3 · Continuously monitor and test	Historical hard-gate evidence is separated from current connector liveness and known bypass gaps.	Coverage states, Incident Pack and explicit disclosures.
4 · Enable end-user responsibility	Operators can inspect authority, decision reasons and evidence strength.	Human-readable exports; training remains a deployer duty.

Why system-level enforcement was selected

The deployment follows MGF v1.5 section 2.3.1, which recommends deterministic safeguards operating at system level, especially for higher-risk actions. The model may recommend or request an action. The protected executor relies on cryptographic authority and exact scope checks, not on the model remembering the restriction.

Contribution to future guidance

The case suggests that organisations should report four evidence levels separately: policy declared, decision recorded, execution boundary enforced and external effect reconciled. This prevents a prompt, dashboard, signed decision and fail-closed executor from being described as equivalent guardrails.

PROPOSED MGF REFINEMENT	Ask each consequential-action control to state its protected path, release condition, failure mode, freshness, bypass scope and strongest evidence level.

3.5 · INDEPENDENT VERIFICATION

A verifier can check integrity without trusting the model or receiving an unrestricted tenant credential.

Fidacy decision and report signatures use EdDSA with Ed25519 keys. Public keys are exposed as a JSON Web Key Set. Signature validity establishes that the payload was issued under the corresponding private key and has not changed since issuance.

Verification procedure

- Fetch the current public keys from https://api.fidacy.com/.well-known/jwks.json.
- Select the key identified by the JWS protected header and require EdDSA.
- Verify the compact signature, issuer, audience, issued-at and expiry claims.
- Compare action, resource and contextHash with the live request.
- Require successful atomic redemption immediately before the connected side effect.
- When completion is claimed, reconcile the authority evidence with the downstream operational receipt.

What signature validity does not prove

Evidence object	Proves	Does not prove alone
Signed decision	Issuer attribution and payload integrity.	Executor compliance or effect completion.
Redeemed grant	Connected executor accepted the exact grant before continuing.	That a PSP, ERP or CRM completed the operation.
Content digest	Later integrity comparison when original content is available.	The nature or meaning of undisclosed content.
Incident Pack	Grouped evidence and declared correlation method.	A cryptographic link where only time-window correlation exists.

EPISTEMIC RULE	Never collapse monitoring, authorisation, enforcement and effect completion into one claim.

3.6 · LIMITATIONS AND LEARNINGS

The strength of the case is the precision of both its proof and its limits.

Demonstrated in the attached assurance run

- Model-independent evaluation of an exact action, resource and local context digest.
- Signed ALLOW and DENY evidence using Ed25519.
- Request-bound release and single-use replay refusal before effect invocation.
- Bounded action budget and delegation that cannot widen parent authority.
- Production out-of-scope DENY, tamper refusal, exact Stripe test refund, replay refusal and Incident Pack export.
- Closed-tenant fail-closed behaviour and live public verification interfaces.

Scope boundaries

- This is a Fidacy-operated reference deployment, not an external customer case, and it moves no live customer funds.
- Enforcement claims apply to Fidacy-connected executor paths; unconnected tools remain outside the proven boundary.
- The downstream effect claim is limited to the Stripe test-mode receipt observed in this case.
- The authority control does not by itself assess model fairness, bias, hallucination, overall task correctness or legal compliance.
- Alignment does not imply IMDA certification, endorsement or regulatory approval.

Key learning

Agentic governance becomes operational when authority is a prerequisite on the action path, not a suggestion inside the model. Assurance becomes credible when each object states exactly whether it proves policy, decision, enforcement or external completion.

OFFER TO IMDA	Fidacy can provide the machine-readable evidence bundle, source-level test vectors and a live production-boundary demonstration using an isolated tenant and Stripe test mode.

REFERENCES AND CONTACT

Submission package and primary sources

1. IMDA Model AI Governance Framework for Agentic AI v1.5

2. Fidacy technical implementation note

3. Action Authority documentation

4. Control Coverage documentation

5. Public verification keys

6. Embedded machine-readable assurance run: Fidacy-IMDA-Assurance-Run-2026-08-29.json

Recommended form attachment

Upload this consolidated PDF under 5. Case study contribution. The complete assurance-run JSON is embedded inside the PDF and can be extracted without changing its bytes. The separate feedback PDF belongs under Feedback on MGF for Agentic AI.

Contact

Lucas de Lima
Founder, Fidacy
lucas@fidacy.com
https://fidacy.com

SUBMISSION DECLARATION	This document describes a Fidacy-operated technical reference deployment. It identifies material limitations and does not claim an external customer deployment, certification or official endorsement.

APPENDIX A · MACHINE-READABLE ASSURANCE RUN

Verbatim evidence bundle

The complete JSON evidence file is reproduced below as visible, selectable text. The identical original file is also embedded in this PDF for direct extraction.

INTEGRITY	Embedded file · Fidacy-IMDA-Assurance-Run-2026-08-29.json · SHA-256 df2994dbba9f4d277fe272b05ae893918b8b374725c7d681692df5d3f2dad46a


```
{
  "format": "fidacy.assurance-run.v2",
  "run_id": "imda-mgf-2026-08-29-30cb4734",
  "executed_at": "2026-08-29",
  "repository": "lucaslubi/fidacy",
  "source_revision": "4fd4748",
  "acceptance_script": "apps/admin/scripts/imda-authority-production-e2e.ts",
  "acceptance_script_sha256":
    "8714b70dc683ad533682407d81fadcd70dd7adb055251e4a72d8e22cf72992d7",
  "purpose": "Reproducible evidence supporting the Fidacy-operated IMDA MGF for Agentic AI
    case study contribution",
  "environment": {
    "component_tests": "local clean-process execution against repository code",
    "production_acceptance": "isolated technical Enterprise tenant against the production
      authority service and Stripe test mode",
    "public_interfaces": "production public endpoints",
    "customer_data_used": false,
    "live_customer_funds_moved": false
  },
  "observations": [
    {
      "control": "Exact action, resource and context binding",
      "method": "packages/firewall/test/action-mandate.test.ts and
        action-grant-jws.test.ts",
      "result": "PASS",
      "evidence": "A valid Ed25519-signed grant verified for its intended request and was
        rejected after context, resource or issuer mismatch."
    },
    {
      "control": "Single-use release",
      "method": "HostedActionGrantEnforcingExecutor replay test",
      "result": "PASS",
      "evidence": "The first eligible execution invoked the side effect once. A second
        executor refused the same grant with grant_replayed; the side effect count remained
        one."
    },
    {
      "control": "Bounded action budget and constrained delegation",
      "method": "DevActionMandateCore policy tests",
      "result": "PASS",
      "evidence": "Budget exhaustion, resource mismatch and attempts to widen actions,
        action budget or delegation depth were rejected."
    },
    {
      "control": "Signed ALLOW and DENY evidence",
      "method": "Ed25519 receipt verification tests",
```

APPENDIX A · CONTINUED · 02

Machine-readable assurance run


```
      "result": "PASS",
      "evidence": "Both ALLOW and DENY receipts verified under the generated public key;
        DENY issued no grant."
    },
    {
      "control": "Public service health",
      "method": "Unauthenticated HTTP GET",
      "result": "PASS",
      "evidence": "https://api.fidacy.com/health and https://api.fidacy.com/ready returned
        HTTP 200 during this run."
    },
    {
      "control": "Independent signature verification interface",
      "method": "Unauthenticated HTTP GET",
      "result": "PASS",
      "evidence": "https://api.fidacy.com/.well-known/jwks.json returned HTTP 200 with two
        EdDSA OKP public keys during this run."
    },
    {
      "control": "Out-of-scope action denial in production",
      "method": "Authenticated production Action Demo through an isolated Enterprise test
        tenant",
      "result": "PASS",
      "evidence": "An attempted export of all customer records returned DENY with signed
        evidence and no executable release."
    },
    {
      "control": "Tampered request refusal before the payment rail",
      "method": "Change the authorised refund from USD 24 to USD 240 while presenting the
        original signed grant",
      "result": "PASS",
      "evidence": "The executor returned REFUSED with context_mismatch and did not call
        Stripe for the altered request."
    },
    {
      "control": "Authorised downstream execution",
      "method": "Redeem the exact request-bound grant and execute the refund through Stripe
        test mode",
      "result": "PASS",
      "evidence": "The exact USD 24 refund returned EXECUTED against a succeeded Stripe
        test-mode payment."
    },
    {
      "control": "Durable replay refusal",
      "method": "Present the same redeemed grant to the executor a second time",
      "result": "PASS",
      "evidence": "The executor returned REFUSED with grant_replayed and did not issue a
        second refund."
    },
    {
      "control": "Reviewable incident evidence",
      "method": "Export the Incident Pack for the executed decision",
      "result": "PASS",
      "evidence": "The pack contained the ALLOW decision, signed receipt and SHA-256
        evidence digest."
    },
    {
      "control": "Closed-tenant fail-closed behaviour",
      "method": "Close the isolated tenant and retry its API key",
      "result": "PASS",
      "evidence": "The tenant was closed, its keys were revoked and the production API
        rejected the former credential."
    }
  ],
  "test_summary": {
    "component_test_files_passed": 2,
    "component_tests_passed": 6,
    "production_acceptance_checks_passed": 9,
    "failed_checks": 0,
    "duration_ms_reported_by_vitest": 433
```

APPENDIX A · CONTINUED · 03

Machine-readable assurance run


```
  },
  "production_evidence": {
    "technical_tenant_id": "c847694b-0ed5-4eea-9de8-1e6e7b305c31",
    "tenant_final_status": "closed",
    "authority_run_id": "f7cc9429-18da-42d0-8a94-db0606beb0ad",
    "incident_evidence_sha256":
      "487c78f826caf4507598d163a9014918a16f885ebc4a4d48eb8c8ea749433831",
    "stripe_mode": "test",
    "refund_amount_minor_units": 2400,
    "refund_currency": "usd"
  },
  "limits": [
    "No independent customer production deployment is claimed.",
    "No live customer funds or customer data were used.",
    "The evidence applies to the Fidacy-connected executor path and does not imply
      visibility into unconnected tools.",
    "IMDA certification, endorsement or regulatory approval is not claimed."
  ],
  "reproduction": [
    "pnpm --filter @fidacy/firewall test -- action-mandate.test.ts
      action-grant-jws.test.ts",
    "NODE_OPTIONS=--conditions=react-server FIDACY_ENGINE_URL=https://api.fidacy.com pnpm
      exec vercel env run -e production -- pnpm exec tsx
      scripts/imda-authority-production-e2e.ts",
    "GET https://api.fidacy.com/health",
    "GET https://api.fidacy.com/ready",
    "GET https://api.fidacy.com/.well-known/jwks.json"
  ]
}
```