# An action is born scoped

The companion story, mapped through Actions Science

## Authority, permission, target enforcement and evidence stay connected to the same action.

> Every system tells three stories: what was allowed, what was tried and what actually happened. Actions Science keeps them apart, reconciles them and shows where they don’t match.

# Deed, Gate, Chain

Three responsibilities across one governed action

### DEED — Who has the right?

Authority and limits for this exact action.

### GATE — May this action happen?

Decide before the effect. The guard enforces it at the target.

### CHAIN — Can another person check it?

Evidence that survives for independent review.

CHAIN retains authority and decisions before the effect, then outcomes and later changes.

# One exact action

The room-finding release used in the companion

| Element | Value |
| --- | --- |
| **Content** | Fixed build 1.1 |
| **Target** | Production |
| **Prepared by** | Production's guard, with a reference number |
| **Use** | One permitted use |
| **Time** | Before expiry |
| **Authority** | Owner approval for 1.1 |

> Build 1.2 has different bytes and needs a new authorization. Approval for build 1.1 cannot authorize build 1.2.

# One action, five stages

Responsibilities run across. The action progresses down.

| Stage | Existing systems | Governed action | CHAIN evidence |
|---|---|---|---|
| 1 Define | Model, data, code checks | Production's guard records build 1.1 | Action identity and reference |
| 2 Establish authority | Owners, roles, approvals | DEED: establish the right | Authority and policy versions |
| 3 Decide | Policy engines | GATE: accept or refuse the release | Permission or refusal |
| 4 Enforce | Target controls | Match the pass to the prepared release | Confirmed result or refusal |
| 5 Recheck | Independent reviewer | Recompute offline; match what was allowed, tried and done; keep any mismatch open | Decision and history checked |

A changed build stops, and the failed check restricts that route. Under the method, CHAIN retains the refusal and evidence at every stage; sealing refusals is designed and tested in RSBIS, with coding scheduled (W190).

# Permission meets the target

Production's guard writes down the exact build, then checks the pass against it.

Production's guard first writes down build 1.1 with a reference number. Gate decision on that prepared release: build 1.1, owner, Production → permit; build 1.2 → refuse.

The guard matches the pass to the release it prepared and checks presenter, expiry, use and that nothing the approval relied on has changed. It deploys once; done only when Production confirms.

- The preparation, the match and that check are designed and tested, with coding scheduled;
- Production's confirmation is designed and tested, with coding scheduled too.

Under the method, a failed check restricts that route: until the owner reviews it, any new pass for that presenter needs the owner's fresh approval.

Close or name every route that can deploy, and test it. A deploy made outside the path is found in Production's own log, as far as that log records it, and restricts the routes it used. Finding it afterward never counts as prevention.

A pass is exact, single-use, short-lived and bound to its presenter, so a compromised actor or a stolen pass can do at most the one approved action, once, before it expires.

# Independent review after the action

The gate can stop. The evidence remains.

### Before the effect

Build 1.1, authority, policy and permission or refusal.

### At the target

Observed result: done only when the target confirms; otherwise refused, failed or interrupted, or unknown. A deploy with no pass is done outside the path, recorded as an incident.

### Through later change

Build 1.2 and later changes, revocations and history, carried in the continuity bundle.

An independent reviewer recomputes the decision and authority offline from portable evidence and the continuity bundle, and matches what was allowed, tried and done.

# Two ways to connect the existing stack

Actions Science applies the same method and tests to either architecture.

**Existing capabilities:** identity and access · models and data · apps and workflows · target systems · audit and archives.

**RSBIS:** Deed, Gate and Chain bind the action across the integrated controls.

**Complete construction:** mature components plus a complete binding architecture of their own.

Qualification depends on the whole path.

# Six views of the whole journey

The six Actions Science pillars

| Pillar | Question |
|---|---|
| **Semantics** | What exactly changes? |
| **Authority** | Who may authorize it, under which limits? |
| **Execution** | Where is permission enforced before the effect? |
| **Evidence** | What can another person verify? |
| **Continuity** | What happens during failure, delay or change? |
| **Assurance** | How strong is the finding, and what challenges it? |

> Each pillar applies across the action stages.

# Requirements that become reusable

The intended operating model for classes of actions

| Level | Holds | Room-finding example |
|---|---|---|
| 01 **Action class** | Shared requirements and challenge sets | Production release |
| 02 **Deployment** | Named owners, target, credentials and routes | Confirm the actual controls and boundaries |
| 03 **Each action** | Current facts and a request with its own permission | Apply class requirements to this attempt |

Relevant changes reopen the affected requirements and tests.

# The Actions Science working cycle

Teams build and refine an action record together.

| Step | What it does |
|---|---|
| 1 Define | Name the exact effect and declared surface. |
| 2 Hypothesize | State what must hold and how to disprove it. |
| 3 Prepare | Assign owners. Gather the needed evidence. |
| 4 Challenge | Try bypass, substitution, replay and change. |
| 5 Judge | Record supported claims, gaps and contradictions. |
| 6 Improve | Fix the mechanism. Refine requirements. |

> The cycle produces bounded findings and reusable requirements.

# Six tests of the complete action path

Tests 1 to 3: authorization and execution

| Test | What it asks |
|---|---|
| **T1 Decision before action**<br>formal: Pre-action decision and rejection | Make a deterministic decision before the effect. Refuse unauthorized actions and retain the refusal. |
| **T2 Rules as they were**<br>formal: Historical policy and authority | Reconstruct the rules and legitimate authority that applied at that time. |
| **T3 Pass tied to the action**<br>formal: Decision-bound execution | The pass for build 1.1 cannot authorize build 1.2, another target or a replay. |

> The tests assess the integrated path and its evidence.

# Six tests of the complete action path

Tests 4 to 6: coverage and independent verification

| Test | What it asks |
|---|---|
| **T4 No way around**<br>formal: Mediation assurance | Close or name every declared route to the effect, and test it. A failed check on a route restricts that route. An effect made outside the path is found in the target's own log, as far as that log records it and whoever went around it cannot change it (reads, changes put back before the next check and effects in other systems are not found this way). Finding it never excuses an open route, and finding it afterward never counts as prevention. |
| **T5 Checkable without you**<br>formal: Operator-independent replay | Recompute the decision offline from retained evidence, the continuity bundle and independently established trust inputs, after keys, operators or systems have changed. |
| **T6 One story throughout**<br>formal: End-to-end semantic coherence | Keep identity, authority, policy, permission, effect and verification tied to the same event. |

> Acceptance is bounded to the tested deployment, surface and trust assumptions.