

# Agentic Process Governance

Robert Betts  
Martyn Riley  
Jamie Riley-Betts

Riley Betts Ltd  
`robert.betts@rileybetts.ai`

5 October 2026

Riley Betts Ltd working paper

## Abstract

When an agent may take a different path to the same goal, the word governance is asked to cover too many different things: a log, a person placed on every step, a label of assurance, a check before an action, and a file that has passed a familiar sequence of review steps. Those are real pieces of discrete work. They are not the same object.

This paper states the governed object. Governance is the maintained relationship among the governing goal, the local obligations the goal depends on, the trace of gated actions, the execution boundary, the proof-support set, and the trust that remains. Write  $P(G_a, x_i \mid A_i)$  for the claim that this trace satisfies this goal relative to the execution boundary  $A_i$ . The proof inside discharge is the different claim  $P(G_a, x_i \mid A_g^i)$ , relative to the proof-support set. The same goal may be met by different traces. Two cases that look like the same journey can still be different claims. Agreement of those obligations on a shared fact is not part of that claim (§7).

The institution pins the goal and the mandatory floor before the run: it fixes and versions them, and the agent cannot edit them. A further requirement, stated here and not derived, is that the party that selects the path does not select the floor. The production of evidence and the gate's decision stay separate. A person who prepares a document has traditionally stood apart from the role that accepts it. An agent can call a tool and receive, in the same run, the evidence the case will rely on. The tool's return does not settle the gate. An *allow*, the gate's decision to permit an action before it executes, is a control on one action. It is not, by itself, the assurance claim. Discharge asks for the proof, coverage of the floor, an acceptable residue, and a support that sits inside the execution boundary. One pinned policy is compiled twice, into the machine that allows or denies a consequential action and into the skill the model reads when it proposes what to do next. That double compilation is a constraint this paper states and does not establish. The mortgage application is the illustration. A fall in income can leave the outcome with a person after the predicates a file can prove have been proved.

The contribution is the separation of an allow from discharge, and the requirement that someone other than the agent pin the floor and the acceptability of residue.

The paper does not supply the rule that chooses between two incomparable sets of assumptions that both cover the floor, and it does not define how two local proofs are shown to use the same shared fact. It does not certify a lending system, and it does not claim that a control has been accepted by an examiner.

**Keywords:** agentic process; governance; proof boundary; trust residue; goal-relative assurance; delegated authority

**How to cite:** Betts, R., Riley, M. and Riley-Betts, J. (2026) 'Agentic Process Governance'. Riley Betts Ltd.

Terms used with a fixed sense in this paper are collected in the glossary before the references. That glossary is the subset of the series master glossary, `master-glossary.md`, that this paper uses. The shared senses are fixed there.

## Contents

|    |                                       |   |
|----|---------------------------------------|---|
| 1  | What the word is being asked to cover | 4 |
| 2  | The governed object                   | 4 |
| 3  | An allow is not the claim             | 5 |
| 4  | Who sets the floor                    | 6 |
| 5  | One policy, two compilations          | 6 |
| 6  | A mortgage application                | 7 |
| 7  | What remains open                     | 8 |
| 8  | What this paper does not claim        | 8 |
| 9  | Glossary                              | 8 |
| 10 | References                            | 9 |

# 1 What the word is being asked to cover

An institution that lets software act has to answer a practical question: if this system can approve, reject, submit, transfer, or otherwise change state, how is that authority governed when the path to the goal is not fixed in advance?

Several answers are already in circulation. A log reconstructs what happened. A person can be asked to approve a step. A vendor can say that a system is assured. A kernel can be asked to prove a proposed action against axioms formalized in advance. A case file can be treated as complete because it has passed a familiar sequence of review steps.

Each of those answers does something. A log is a record. A person on a step is oversight of that step. A pre-execution check is a control on that action. A familiar sequence of review steps is a journey. As in the abstract, governance in this paper means the further work of keeping these things visible together: the governing goal, the local obligations the goal depends on, the trace of gated actions, the execution boundary  $A_i$ , the proof-support set  $A_g^i$ , and the residue that remains. Agreement of those obligations on a shared fact is left open in §7.

The two earlier papers supply the formal part of that work. *Trust and Proof* treats a proof claim as boundary-relative and shows that local discharge of trust does not empty the residue (Betts, 2026a). *Goal-Relative Assurance* moves the unit from a fixed process to a goal, a generated trace, and a boundary that the run may partly construct (Betts, Riley and Riley-Betts, 2026b). This paper does not re-prove those results. It says what they come to when an agent is allowed to act, and what still has to be pinned by someone other than the agent.

## 2 The governed object

For a case of type  $a$ , let  $G_a$  be the governing goal,  $x_i$  the trace of this instance, and  $A_i$  the execution boundary of that trace: the assumptions and dependencies this run relied on. The proof-support set  $A_g^i$  is the set the proof inside discharge is relative to. It sits inside the execution boundary only when hygiene holds. The governed claim is

$$P(G_a, x_i \mid A_i).$$

$G_a$  is fixed for the case type and for the policy version the institution has pinned.  $x_i$  is not. An agent may select evidence, tools, order, and escalation. A second instance can satisfy the same  $G_a$  with a different trace. An identical trace can still sit on a different residue, because  $x_i = x_j$  does not entail  $R(A_g^i) = R(A_g^j)$  (Betts, Riley and Riley-Betts, 2026b).

The local obligations are the predicates the goal depends on. A permitted next action is the gate's decision on one action. A member of the proof support, the assumptions the proof actually used (§3), is not an obligation. The agent is not marked against one fixed order of steps. Where two obligations do not depend on each other, they may be met in either order. Where one depends on another, the second waits. Where the goal allows alternatives, one alternative meets that obligation.

The trace is the append-only chain of actions the gate has allowed, with their parameters and results. A denial is recorded and does not advance the state the next proposal is judged against. Monitors and human review read this chain. They do not read the model's unexecuted plan.

The execution boundary is what the governed claim is relative to. The residue is what still has to be trusted once that proof is granted. §3 says which residue discharge accepts. Governance is the work of naming the boundary, discharging what mechanism can discharge, exposing the residue, and staying answerable when the boundary changes while the visible action does not (Betts, 2026a).

### 3 An allow is not the claim

*Trust and Proof* writes  $P(x \mid A)$  for a required property of  $x$  established relative to assumptions and dependencies  $A$ . The residue  $R(A)$  is the part of  $A$  that still has to be trusted once the rest of that boundary is granted. A local claim can be granted and still leave  $R(A)$  non-empty. That is the Trust Displacement Result. Betts (2026a) treats a smaller residue as a stronger comparison between boundaries. This paper cites that comparison and does not develop it. An empty residue is not required, and it is not implied by a valid proof (Betts, 2026a).

*Goal-Relative Assurance* distinguishes three sets in a generated execution.  $A_i$  is the execution boundary: what the run relied on.  $A_g^i$  is the proof-support set: what the proof actually used.  $A_G$  is the mandatory floor the authority imposes for every acceptable proof of the goal. A proof boundary is the set in the conditioning place of a proof claim. For the proof inside discharge, that set is  $A_g^i$ . For the governed claim, it is  $A_i$ . Covering, written  $A_g^i \models A_G$ , asks the support to meet the floor in substance, not by reusing a requirement’s name. Hygiene, the requirement that the support used sit inside the execution boundary, is separate:  $A_g^i \subseteq A_i$ . It does not follow from the proof or from covering. The governed claim this paper writes is  $P(G_a, x_i \mid A_i)$ . The proof inside discharge is  $P(G_a, x_i \mid A_g^i)$ . That proof is  $P(x \mid A)$  with the goal in the property place and the execution as an argument. Hygiene does not make the two formulas one claim. Boundary sufficiency is covering together with  $P(G_a, x_i \mid A_g^i)$ . Discharge holds only if four conditions hold: that proof, the floor, an acceptable residue  $\text{Acceptable}_G(R(A_g^i))$ , and hygiene. The first three are independent of one another. Hygiene is not part of that independence result. Meeting the four conditions does not by itself constitute discharge. The residue discharge accepts is  $R(A_g^i)$ , the part of the support still trusted. From  $A_g^i \subseteq A_i$  it does not follow that  $R(A_g^i) \subseteq R(A_i)$ . A valid proof does not empty that residue (Betts, Riley and Riley-Betts, 2026b).

A gate sits earlier than that claim. From the current state, the pinned policy says which consequential actions would be allowed next. The harness, the software that runs the model and submits its proposed actions, proposes. The machine allows or denies. On allow, the action may change state, and the allow is appended to the trace. That is a control on one action. It becomes part of  $x_i$ . It does not, by that fact, show the proof, that the support covered  $A_G$ , that the residue  $R(A_g^i)$  is acceptable, or that the support sits inside the execution boundary. An agent can call a tool and receive, in the same run, the evidence the case will rely on. That return does not allow the action. The allow is the gate’s decision against the pinned policy.

The distinction matters because a check before execution is easy to treat as the whole of assurance. Rashie and Rashi (2026) describe one such check, an independent example that did not inform this work. A proposed financial action is submitted to a Lean 4 kernel as a conjecture, and it is allowed to run only if the kernel accepts a proof from regulatory axioms formalized in advance. The preprint states the guarantee as correctness relative to that formalized statement. Discharge still asks for the proof, for the support to meet the institution’s floor in substance, for an acceptable residue, and for the support to sit inside the execution boundary. The floor charge is covering: the support meets  $A_G$ . It is not a second floor standing beside the axioms. Those conditions lie outside the guarantee their paper states.

Proof-carrying code shows that an untrusted producer can supply a machine-checkable argument that an artefact satisfies a stated policy (Necula, 1997). That result stops at the stated policy. A digest or a certificate, separately, can show that a named transition held relative to its circuit, the object the certificate is relative to (Betts, Riley and Riley-Betts, 2026b). This paper’s further point is that satisfying a stated policy, or holding relative to that object, does not decide that the policy or the object is the institution’s  $A_G$ , and does not decide that the residue is acceptable. A floor written as the name of a mechanism can be satisfied by the wrong mechanism. Covering is a relation of substance. A stored hash of a policy object names a mechanism. It does not show that the object is the floor.

## 4 Who sets the floor

$A_G$  and  $\text{Acceptable}_G$  are parameters. The papers do not derive them from the trace (Betts, Riley and Riley-Betts, 2026b). This paper requires that the agent not pin them. A further requirement is stated here and is not derived: the party that selects the path must not also be the party that selects the floor.

Before an instance runs, obligation, institutional interpretation, and the machine-checkable property are different steps. The obligation is  $G_a$ . The interpretation is an authority choosing  $A_G$  for that goal. The machine-checkable property is what the pinned policy can allow or deny. Enforcement is the allow or the denial. The evidence is the gated trace. Outside what mechanism has discharged sit the judgements that were never delegated, together with the residue that remains to be trusted. The residue discharge accepts is part of the support,  $R(A_g^i)$ . It is not the residue of the execution boundary,  $R(A_i)$ .

Authority can be delegated in degrees. A recommendation can remain inside a human decision. An action that changes state has been delegated. At each step an institution can ask the following questions, and this account gives each an answer:

1. *Who holds the authority, and what is handed to the agent?* The institution pins  $G_a$ ,  $A_G$ , and  $\text{Acceptable}_G$  before the run. The closed set of gated actions is what is handed to the agent.
2. *What must be true before it is used, and what does the machine enforce?* An allow is mechanical enforcement of a guard.
3. *What can be formally proved of that machine?* A proved property of the machine is a stronger fact than the allow.
4. *What stays with a person?* An outcome such as review.
5. *Which assumptions remain?* The residue.
6. *What record has to survive?* The trace.

What that authority still does not supply is a selection rule. Two products can see the same evidence and still differ, because  $G_a$  changes with the product, and with it  $A_G$  and  $\text{Acceptable}_G$ . The residue discharge accepts can therefore differ. Two incomparable supports can both cover one floor. Governance has to say which support counts, not only whether some support exists. That rule would be owned with the floor. It is not stated in this paper.

## 5 One policy, two compilations

The same body of obligation, institutional policy, and product criteria is compiled twice. Both compilations are pinned. The agent does not edit either one at runtime.

It is compiled into the policy machine. The machine is a closed alphabet of consequential actions, guarded transitions, and invariants. Every consequential action is proposed to it and allowed or denied by it. What the machine permits from the current snapshot is the process as seen from the current state: the actions that would be allowed next. The machine is the artefact the allow depends on, so it is the artefact that has to be formally specified and proved. That required proof is a stronger fact than the allow. It is not the four-part discharge. A model's confidence, a fluent explanation, and a passage of instructions do not take the place of that proof.

The same source is compiled into the skill the model reads when it proposes the next actions. The skill states the process, the named actions, and the policy in a form the model can use. Those proposals are how the path is chosen. They become part of the trace only after the

machine allows them. If the model improvises, skips, or invents a name, the machine still gates the call.

The two compilations are views of one policy. They agree when every action the skill can propose is an action the machine can gate, and every guard the machine enforces is still visible to the skill. A proposal the machine cannot name, or a guard the skill has dropped, is a failure of that agreement. Writing both artefacts from the same document does not prove the agreement. When the policy changes, a new machine and a new skill are pinned together.

This double compilation is a design constraint this paper states and does not establish. It does not come from Betts (2026a) or from Betts, Riley and Riley-Betts (2026b) or from Rashie and Rashi (2026). It is not evidence about the mortgage case in §6.

## 6 A mortgage application

Someone who has processed applications sees a journey. There is a start, a sequence of stages, and an end. Identity, then income, then affordability, then the property, then a decision. Two files that have been through the application look alike, because each has passed a familiar sequence of review steps. In that picture the process is the stable thing, and success is reaching the end.

$P(G_a, x_i \mid A_i)$  does not take the journey as the object.  $G_a$  is the governing goal for a case of this type under this product's pinned criteria.  $x_i$  is the append-only chain of actions the gate has allowed, with their parameters and results.  $A_i$  is what that run relied on. A second applicant of the same type can satisfy the same goal with a different trace and a different execution boundary. A file that felt unfinished can already have met every applicable obligation. A file that felt complete can still be open, because one predicate of the goal is unresolved. An unresolved predicate of the goal is not, by that fact, a member of  $A_i$ .

The goal is a set of obligations: identity, income, affordability, deposit, and property. They compose in three ways. A final conjunction joins obligations that are otherwise independent. A branch chooses among structures, such as sole trader or limited company. A real dependency orders them, such as affordability waiting until income resolves. Income resolving does not by itself fix the figure affordability uses. That agreement is open in §7. Where there is no dependency, the obligations can be met without a single prescribed order. The case holds the state of each obligation. The rest of the process is whatever is still unresolved.

Criterion, evidence, proof, and outcome sit on different layers. The criterion is part of  $G_a$ . The evidence and the proof sit in the trace and the execution boundary. The outcome, including a referral, is what remains after the claim has been stated. A predicate proved locally can leave the outcome open.

Take a limited-company applicant under a pinned criteria version. The named predicates, the threshold, and the review route are stipulated for this illustration. Identity, residency, loan to value, trading history, shareholding, accounts, and salary-and-dividend income are proved locally: each named predicate holds. That local proof is not covering, not an acceptable residue, and not agreement on the shared income figure. Income is down on the year before, past the threshold the criteria name. The predicates a file can prove have been proved in that local sense. The trend condition stays with the outcome. Its permitted route, under this stipulation, is review by a person, with named evidence still required. The decision to accept, decline, or refer stays with that person. The journey and the assurance claim have come apart.

Two products can see the same evidence and open different residues. That is  $G_a$  changing with the product, and with it  $A_G$  and  $\text{Acceptable}_G$ , while the trace is what this case was allowed to record.

## 7 What remains open

Composition of local proofs is not a final conjunction of pass bits. The income figure shared by the income obligation and the affordability obligation is the example. Agreement would mean that both proofs restrict to the same figure inside one boundary  $A_i$ . Affordability can be satisfied relative to one income figure while the income obligation proved another, and both can still return satisfied. The maps that would send each proof to that shared figure are not defined here. Agreement on a shared fact is not part of the claim this paper makes.

The selection rule of §4 is open in the same sense. The institution has to own it with the floor. This paper does not choose it.

## 8 What this paper does not claim

Goal-oriented requirements already treat a goal as stable while implementations vary (Dardenne, van Lamsweerde and Fickas, 1993; van Lamsweerde, 2001). Declarative process models already permit many traces to satisfy a common constraint set (van der Aalst, Pesic and Schonenberg, 2009). The conjunction, branch, and dependency in §6 are that pattern of composition. The difference this paper states is an allow kept apart from discharge, and a floor pinned by someone other than the agent.

Runtime certification asks how certification evidence can be produced for a run that was not certified in advance (Rushby, 2008). Runtime assurance keeps an untrusted decider to a boundary set outside it. Intervention there keeps a safety property, and it is not proof that the controller met its mission (Schierman et al., 2020). This paper’s claim is a goal-relative discharge: the proof, covering in substance, and an acceptability judgment on the residue are distinct, and hygiene is required separately. The floor and that acceptability are pinned by someone other than the agent.

The idea of residual or minimal trust predates the residue defined in Betts (2026a) (Bella, 2007). This paper does not claim the ideas from those lines of work as its own. Betts, Riley and Riley-Betts (2026b) already record that limit for the goal-relative model. The contribution here is the separation of an allow from discharge, and the requirement that the floor and the acceptability of residue be pinned by someone other than the agent.

The mortgage application is an illustration. It is not a full lending policy, and this paper does not certify a lending system. A formally specified machine checks the relations it is given. It does not decide that a particular floor is the right reading of an obligation.

## 9 Glossary

This glossary is a subset of the series master glossary. Senses fixed in Betts (2026a) and Betts, Riley and Riley-Betts (2026b) are recorded here only as this paper uses them.

**Governing goal**  $G_a$ . The goal for a case of type  $a$ , fixed for that type and for the pinned policy version.

**Trace**  $x_i$ . The append-only chain of actions the gate has allowed, with their parameters and results. A denial is recorded and does not advance the state the next proposal is judged against.

**Execution boundary**  $A_i$ . What this run relied on. The governed claim is relative to this set.

**Proof-support set**  $A_g^i$ . The assumptions the proof actually used.

**Mandatory floor**  $A_G$ . The assurance requirements the authority imposes for every acceptable proof of the goal. The agent does not pin it.

**Governed claim and proof inside discharge.** The governed claim is  $P(G_a, x_i \mid A_i)$ . The proof inside discharge is  $P(G_a, x_i \mid A_g^i)$ . Hygiene does not make those two formulas one claim.

**Covering,  $\models$ .**  $A_g^i \models A_G$ . The support meets the floor in substance, not by reusing a requirement's name. Also called the floor charge.

**Boundary hygiene.**  $A_g^i \subseteq A_i$ . Required for discharge, and not entailed by the proof or by covering.

**Residue.**  $R(A_g^i)$  is the part of the support still trusted, and it is the residue discharge accepts.  $R(A_i)$  is the residue of the execution boundary. Support inclusion does not yield residue inclusion.

**Acceptable<sub>G</sub>.** The parameter that says whether  $R(A_g^i)$  is acceptable. Not derived from the trace. The agent does not pin it.

**Discharge.** Holds only if four conditions hold: the proof  $P(G_a, x_i \mid A_g^i)$ , the floor, an acceptable residue, and hygiene. The first three are independent of one another. Meeting the four conditions does not by itself constitute discharge.

**Trust Displacement Result.** A local claim can be granted and still leave  $R(A)$  non-empty. This paper cites the result and does not develop  $\prec_t$ .

**Allow.** The gate's decision to permit one action before it executes. A control on that action, and part of  $x_i$ . Not the assurance claim.

**Gate.** The decision, from the current state and the pinned policy, on which consequential actions would be allowed next.

**Harness.** The software that runs the model and submits its proposed actions.

**Policy machine.** The closed alphabet of consequential actions, guarded transitions, and invariants. A proved property of the machine is stronger than an allow, and it is not discharge.

**Skill.** The compilation of the same policy into the form the model reads when it proposes the next actions.

**Double compilation.** One pinned policy compiled into the policy machine and into the skill. A design constraint this paper states and does not derive from the other two.

**Local obligations.** The predicates the goal depends on. A member of the proof support is not an obligation. An unresolved predicate of the goal is not, by that fact, a member of  $A_i$ .

## 10 References

- Bella, G. (2007) *Formal Correctness of Security Protocols*. Berlin: Springer. DOI: 10.1007/978-3-540-68136-6.
- Betts, R. (2026a) 'Trust and Proof'. Riley Betts Ltd. Available at: <https://rileybetts.ai/papers/trust-and-proof>
- Betts, R., Riley, M. and Riley-Betts, J. (2026b) 'Goal-Relative Assurance of Dynamically Generated Execution Traces'. Riley Betts Ltd. Available at: <https://rileybetts.ai/papers/goal-relative-assurance>

- Dardenne, A., van Lamsweerde, A. and Fickas, S. (1993) ‘Goal-directed requirements acquisition’, *Science of Computer Programming*, 20(1–2), pp. 3–50. DOI: 10.1016/0167-6423(93)90021-G
- Necula, G.C. (1997) ‘Proof-carrying code’, *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL ’97)*, pp. 106–119. DOI: 10.1145/263699.263712
- Rashie, D. and Rashi, V. (2026) ‘Type-Checked Compliance: Deterministic Guardrails for Agentic Financial Systems Using Lean 4 Theorem Proving’, arXiv:2604.01483. Available at: <https://arxiv.org/abs/2604.01483>
- Rushby, J. (2008) ‘Runtime certification’, in Leucker, M. (ed.) *Runtime Verification: RV 2008*. Lecture Notes in Computer Science, 5289. Berlin: Springer, pp. 21–35. DOI: 10.1007/978-3-540-89247-2\_2
- Schierman, J.D., DeVore, M.D., Richards, N.D. and Clark, M.A. (2020) ‘Runtime assurance for autonomous aerospace systems’, *Journal of Guidance, Control, and Dynamics*, 43(12), pp. 2205–2217. DOI: 10.2514/1.G004862
- van der Aalst, W.M.P., Pesic, M. and Schonenberg, H. (2009) ‘Declarative workflows: balancing between flexibility and support’, *Computer Science — Research and Development*, 23(2), pp. 99–113. DOI: 10.1007/s00450-009-0057-9
- van Lamsweerde, A. (2001) ‘Goal-oriented requirements engineering: a guided tour’, *Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE 2001)*, pp. 249–263. DOI: 10.1109/ISRE.2001.948567