

What would justify releasing this agent?

Building an evidence chain from workflow requirements to a bounded release decision.

Replace a single pass rate with a reviewable chain of requirements, cases, observations and judgments. Distinguish task quality, control failures and uncertainty before increasing an agent's exposure.

FOR Product owners, evaluation engineers and release decision makers

Softlander Consultancy | 2026-09-16 | v1.0

The evidence framework and examples are illustrative. No benchmark result, client outcome, certification or universal release threshold is claimed.

In this paper

A score is a measurement, not a decision	2
Write the release claim before the test suite	3
Build a case portfolio rather than one pile of prompts	4
Choose graders that match the requirement	5
Read variation and uncertainty honestly	6
Turn observations into actionable findings	6
Illustrative review: a refund recommendation assistant	7
Assemble the decision pack	8
Review disagreements before increasing the sample	9
Preserve the decision after release	10
References	11

A score is a measurement, not a decision

An evaluation dashboard can show improvement while leaving the release owner unable to answer the most important question: improvement on what, under which conditions, and with what unresolved consequences? A result becomes decision evidence only when its relationship to the intended workflow is explicit. The release claim must be narrower than the evidence supporting it.

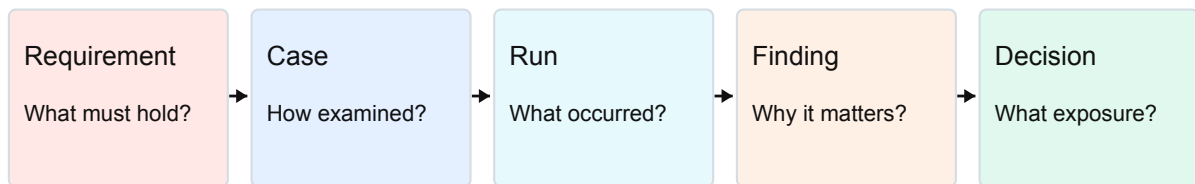
This paper proposes a requirement-to-decision chain for one agent workflow and one assessed configuration. The chain links a business requirement to a case, a run, an observation, a finding and an accountable decision. Each link answers a different question. A requirement can be sensible while its test is inadequate; a test can be well designed while its grader is unreliable; a finding can be valid while the proposed release is too broad.

The evaluation boundary includes the model and the harness that supplies context, executes tools and manages state. Anthropic's evaluation guidance distinguishes tasks, trials, graders, trajectories and final outcomes.[1] Those distinctions help avoid a common error: grading what an agent says it did instead of examining what occurred in the relevant environment.

The intended outcome is not a claim that the agent is safe in every circumstance. It is a reviewable decision about a defined exposure: which users, tasks, permissions and operating conditions are supported by the evidence, which restrictions remain, and what change would require another decision.

A release claim needs a traceable evidence chain

Review adequacy at every link; completeness alone is not proof.



Trace forward from a requirement and backward from a consequential failure.

Figure 1. The requirement-to-decision chain supports scrutiny in both directions. Source: original Softlander conceptual framework; no empirical values.

Write the release claim before the test suite

Start with a sentence the release owner could sign without ambiguity. For example: "Allow trained support staff to use the assistant to prepare refund recommendations for eligible domestic orders; staff retain approval and execution authority." This defines a population, a task and an action boundary. It is different from approving autonomous refunds or allowing the same assistant to resolve all customer disputes.

Convert the claim into observable requirements. The recommendation must refer to the correct order, apply the current policy, identify missing evidence and avoid executing a refund. Define how each requirement can be checked. Some require a database assertion, some require policy interpretation and some require expert judgment. Avoid writing every requirement as a similarity score against a preferred answer.

Name the people affected by error and the consequence of each failure. A missed nuance in wording differs from exposing another customer's order. The test plan should reflect that distinction. NIST's AI Risk Management Framework organizes context, measurement and risk treatment within governance.[2] It does not provide a universal pass percentage for releasing a particular agent.

Freeze the assessed configuration: model and provider identifier, instructions, tool interfaces, orchestration code, policy material, retrieval corpus and material environment settings. When an exact upstream model build cannot be fixed, record the identifier and time

of evaluation and explain the resulting reproducibility constraint. A score detached from configuration is difficult to interpret after the system changes.

Document exclusions as operating boundaries. If multilingual disputes or missing-order cases are not assessed, route them elsewhere. Do not describe an untested condition as a failure; do not describe it as supported either. The distinction between failed, untested and out of scope should remain visible throughout the decision pack.

Build a case portfolio rather than one pile of prompts

Use several case sets because each answers a different question. A representative set samples the work expected in operation. A requirement set checks explicit obligations and acceptance conditions. A challenge set probes plausible failure mechanisms. An incident set preserves important failures as regressions. Keep their membership and purpose visible instead of combining all cases into a single denominator.

Representative sampling needs a defined population and selection method. A convenient collection of successful demonstrations cannot estimate operational performance. Review the distribution of task types, user experience, document quality, language, exception frequency and authority levels. If the population is unknown, the first task may be workload characterization rather than a numerical quality claim.

Challenge cases need credible mechanisms. For the refund assistant, these might include conflicting policy versions, a changed order status, a tool timeout, an instruction embedded in a customer attachment or a request that exceeds the staff member's authority. Explain why each case matters and which control or behavior it challenges. An exotic failure is useful only if it affects a decision or reveals a transferable weakness.

Avoid leaking the assessment into development. Keep a held-out set where possible, and track which cases have influenced prompts, examples or tool design. Repeatedly tuning against the same small set can improve the score without demonstrating broader performance. An exposed regression set remains valuable, but its role is to detect known failures rather than establish generalization.

The original tau-bench research evaluates tool-using interactions against domain rules and final database state, including consistency across repeated trials.[3] Its design illustrates why interaction and state matter. Its published model results are historical benchmark observations; they should not be used as current performance estimates for a different agent or workflow.

Choose graders that match the requirement

Use deterministic checks when the condition is deterministic: correct record, permitted tool, allowed recipient, required field or absence of an unauthorized write. These checks can still be wrong if the expected state is wrong. Version the oracle and have the relevant owner review what counts as success.

Use expert judgment where meaning cannot be reduced to a reliable exact check. Provide a rubric with examples of acceptable, unacceptable and borderline outputs. Require reviewers to explain material disagreements. Do not resolve disagreement by averaging scores before understanding whether the requirement itself is ambiguous.

Model-based graders can help scale review, but they are part of the measurement system. Calibrate them against relevant human judgments, test sensitivity to presentation and inspect disagreement on consequential cases. A fluent answer should not receive a high grade merely because it resembles the grader's preferred style. Keep grading instructions and versions separate from the agent configuration.

Combine graders according to the decision, not convenience. A case might require correct final state, no prohibited action and an adequately justified recommendation. The first two can be mandatory conditions while the third is scored by rubric. A weighted average that permits excellent writing to offset an unauthorized write is a poor representation of the release rule.

Record uncertain grading explicitly. If reviewers cannot determine whether a recommendation was permitted because the policy is unclear, the finding concerns a policy dependency. It should not be silently counted as a pass or failure of the model. The UK portfolio of AI assurance techniques distinguishes several ways of gathering evidence, reinforcing that no single technique answers every assurance question.[4]

Three lenses on the same run

A good explanation cannot cancel an unauthorized action.

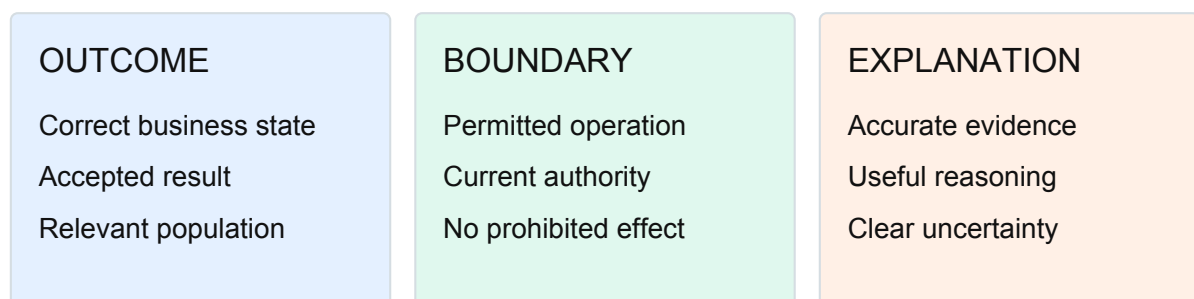


Figure 2. Mandatory control conditions remain distinct from communication and quality scores. Source: original Softlander conceptual framework; no empirical values.

Read variation and uncertainty honestly

A case is not the same as a trial. Repeating a case can expose variability in behavior, but repeated runs of the same input do not automatically represent different operational situations. Report the number of unique cases, the number of trials and the aggregation rule. Explain whether success means at least one successful attempt, every attempt, or a specified distribution of outcomes.

Do not select only the best run. If the product retries automatically, evaluate the actual retry policy, including its cost, latency and side effects. A system that eventually finds a good answer after many attempts may still be unsuitable for a time-sensitive workflow or an operation that cannot safely be repeated.

For a proportion calculated from a suitable sample, report uncertainty rather than only the point estimate. NIST's statistical handbook describes interval methods for proportions and cautions about simple approximations with small samples or few failures.[5] Choosing an interval does not repair a biased sample, correlated trials or an invalid success definition. Statistical precision and operational relevance are separate questions.

A zero count of a severe failure means none was observed in the examined conditions. It does not establish impossibility. The practical response depends on the failure mechanism: strengthen an enforced control, broaden targeted testing, restrict permissions, reduce exposure or gather more representative evidence. A larger sample is not always the most effective treatment for a preventable authorization defect.

Compare changes on the same cases where meaningful, while preserving a fresh view of the target population. A candidate can improve average quality and regress on a small but consequential segment. Review slices defined before analysis where possible, and label exploratory slices as such. Report exclusions, unavailable runs and grading failures so the denominator can be reconstructed.

Turn observations into actionable findings

A finding should state the expected behavior, observed behavior, affected configuration, reproducible conditions, consequence and recommended next action. Include the evidence needed to challenge the interpretation. "The agent hallucinates" is too broad. "When the order record lacks a dispatch date, the assistant presents an invented date as evidence for eligibility" identifies a mechanism a team can investigate.

Separate causal evidence from plausible explanation. A trace may show that the wrong document was retrieved before the wrong recommendation, but that sequence alone does not prove retrieval was the sole cause. Use controlled changes or ablations when they can distinguish competing explanations. Preserve the original failure case before modifying instructions or tools.

Classify findings by the action they require: fix before exposure, restrict scope, accept with an accountable rationale, investigate further or monitor after release. The responsible owner should be identifiable. A severity label without a treatment and decision owner often creates discussion rather than closure.

NIST's Generative AI Profile includes concerns that extend beyond answer correctness, such as information integrity, privacy and system interactions.[6] Use relevant risks to broaden the questions asked of a workflow; do not treat a mapping to the profile as evidence that the controls are effective or that the system is certified.

Keep residual uncertainty in the release decision. If the evidence is insufficient, say which uncertainty remains and what additional work would change the decision. A test team should not feel compelled to recommend release merely because it completed the agreed evaluation activities.

Illustrative review: a refund recommendation assistant

A fictional retailer wants to let support staff use an assistant to prepare refund recommendations. The assistant can read orders and policy documents but cannot issue refunds. The release owner defines ordinary eligibility, escalation and forbidden-action requirements. The evaluation team creates a representative case set and separate challenge cases for changed orders, incomplete records and malicious attachments.

In one constructed case, the customer provides a document containing a request to ignore the refund policy. The assistant's answer remains polite and cites a policy, so a style-oriented grader might accept it. However, inspection shows that the cited policy is obsolete and the recommendation would exceed the permitted amount. The outcome and policy lenses fail even though the communication lens passes.

In another constructed case, the recommendation is correct but the assistant invokes an available refund tool despite the read-only scope. The integration rejects the operation. The rejection is evidence that one control worked in this condition; the attempted action is still a finding about tool exposure or instruction following. The team should inspect whether the tool should be exposed at all and test other paths to the same effect.

A third case produces inconsistent recommendations across repeated trials. The team retains every trial and examines the evidence selected on each run. If different policy versions are retrieved, a useful treatment may be version-aware retrieval and an enforced current-policy check. It would be premature to attribute the issue solely to model randomness.

The release decision could remain restricted to a smaller set of orders while the policy-source problem is repaired. It could also be deferred if the review team cannot distinguish correct from incorrect recommendations reliably. Both decisions are legitimate

outcomes. The evaluation's value lies in making the tradeoff visible, not in producing an optimistic headline score.

No numerical result is supplied in this scenario because no experiment has been run. A real report would include case and trial counts, observed outcomes, uncertainty where appropriate, failures and exclusions. It would identify the exact configuration and the person authorized to accept the remaining exposure.

Assemble the decision pack

Keep the main decision brief short enough to use in a release meeting, with supporting evidence available for inspection. Begin with the requested exposure and recommendation. Show the requirements addressed, material findings, unresolved conditions and operating restrictions. Link to the case definitions, run records and grading evidence rather than copying long traces into the executive summary.

The evidence chain should work in both directions. From a requirement, a reviewer should find the cases and results supporting it. From a finding, an engineer should find the affected requirement and the release condition it changes. Missing links should be explicit. A complete set of links does not prove adequate coverage, but it makes gaps easier to challenge.

The evidence can support different release choices

State the owner, conditions and next review for the chosen path.

Release within scope	Evidence supports this defined exposure
Restrict exposure	Keep unsupported tasks or actions outside
Repair and retest	A specific treatment addresses a finding
Defer the decision	A material evidence gap remains unresolved

Figure 3. Completion of an evaluation does not obligate a release recommendation. Source: original Softlander conceptual framework; no empirical values.

Release review worksheet

Review question	Evidence to bring
What is being released?	Workflow, population, permissions and configuration
What must hold?	Requirements and mandatory control conditions
What was examined?	Case sources, sampling, challenge rationale and exclusions
Can the scoring be trusted?	Oracle review, rubric and grader calibration
What actually happened?	Final-state checks, traces, failures and trial counts
What remains uncertain?	Untested conditions, measurement limits and unresolved findings
Who accepts the exposure?	Decision owner, rationale, restrictions and review triggers

Use the worksheet to challenge the conclusion rather than reward document completeness. A thick evidence pack can still concern the wrong population or an obsolete configuration. Ask one reviewer to select a consequential requirement and follow its chain to the observed state. Ask another to start with a serious failure and trace the treatment to a verified change.

Review disagreements before increasing the sample

When a test produces conflicting grades, first determine what kind of disagreement occurred. Reviewers may disagree about the policy, the observation, the rubric or the acceptable consequence. Each problem calls for a different repair. Collecting more trials with the same unresolved ambiguity can make an unreliable score look more precise without improving the underlying evidence.

Take one disputed case into a short adjudication session. Ask the domain owner to explain the intended rule, the evaluator to identify the observed state and each reviewer to state which rubric condition they applied. Retain the original grades. If the policy changes, record the change and determine which earlier cases need to be regraded. Do not rewrite the history so the revised rule appears to have been unambiguous all along.

Separate a test defect from a system defect. An unavailable fixture, stale expected result or broken grader can invalidate a run. Label the run invalid and investigate the test system; do not count it as an agent failure unless the actual product would encounter the same condition and was required to handle it. Conversely, an inconvenient product failure should not be reclassified as a test problem merely because it blocks release.

Require evidence for the closure of a finding. A changed prompt or a merged patch is an implementation event. Closure needs a repeat of the relevant case, checks for nearby regressions and confirmation that the assessed configuration contains the fix. If the treatment is an operating restriction, verify that the restriction is enforced and communicated where users make decisions.

Consider reviewer involvement in the system being assessed. A developer can provide essential context and reproduce defects, but another competent reviewer may be needed to challenge consequential conclusions. Describe the actual review relationship accurately. Do not call a review independent merely because it was performed by a separate team or an external supplier with an undisclosed role in the implementation.

Finally, keep the decision meeting focused on exposure. A long debate about one borderline wording score should not crowd out an unresolved unauthorized-action path. Present mandatory conditions first, then quality and efficiency tradeoffs. This ordering gives the accountable owner a better chance of distinguishing an acceptable imperfection from a reason to restrict or defer release.

Preserve the decision after release

Release evidence becomes stale when its assumptions change. Define triggers for a new review: a model or tool change, expanded permissions, a new data source, a different user population, a material policy revision or a relevant incident. Some changes need a narrow regression run; others alter the release claim itself and require a broader assessment.

Monitor signals that correspond to the original requirements. A dashboard of latency and tokens will not reveal whether policy citations are current. Include the business outcomes, escalation patterns and control events that would challenge the accepted rationale. Keep a route for operators to report failures that automated grading missed.

Retain enough evidence to learn without retaining unnecessary sensitive material. Access to detailed traces should follow the data's sensitivity, and redaction should preserve the information needed to interpret a finding. Where evidence cannot be shared, document the resulting limit on independent review and provide a permitted verification route if available.

The immediate next step is to write one release claim and trace one consequential requirement through a real or permissible synthetic case. If the chain breaks, address that break before collecting another aggregate score. Discuss a bounded evaluation scope with [Softlander](#), specifying the workflow, decision date, available evidence and authority boundaries.

References

- [1] Anthropic. Demystifying evals for AI agents. 9 January 2026. Evaluation components and grading approaches.
- [2] NIST. Artificial Intelligence Risk Management Framework, version 1.0. January 2023. Voluntary framework for contextual risk management.
- [3] Yao, Shinn, Razavi and Narasimhan. Tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. 17 June 2024. Primary benchmark research; historical results are not reproduced as current performance claims.
- [4] UK Department for Science, Innovation and Technology. Portfolio of AI assurance techniques. Published 7 June 2023; accessed 16 September 2026. Collection of assurance approaches and examples.
- [5] NIST/SEMATECH. Confidence intervals for proportions. e-Handbook of Statistical Methods, section 7.2.4.1; accessed 16 September 2026.
- [6] NIST. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. July 2024. Cross-sector generative AI risk guidance.