SG1 CONSULTING

WHITE PAPER

REVERSIBILITY

Rollback & Reversibility in Automated Workflows

A buyer's guide to testing what a platform can actually undo — the vocabulary that separates real reversal from a support ticket, and a repeatable test that forces a demonstration instead of a claim.

For compliance, vendor-risk, procurement, and legal-ops evaluators

The short version

"Rollback" is not one feature. It is a ladder that runs from a true per-action undo at the top, down through a coarse restore of a whole dataset, to a person promising to fix it later. Most vendor claims sit several rungs lower than the buyer assumes — and the gap is invisible until the day you need to reverse exactly one automated step without disturbing anything else.

"Do you support rollback?" should never end a conversation; it should start one. This paper gives a compliance, procurement, or legal-ops evaluator a precise vocabulary for reversibility, the mechanisms that make it real rather than theatrical, the line past which nothing can truly be undone, the structural properties that make a reversal claim trustworthy, a weighted scorecard, and a hands-on acceptance test that forces any platform to *demonstrate* what it can undo — in what order, within what window, and whether the record can be trusted afterward.

1. Rollback is a ladder, not a checkbox

The single question that separates a real reversal from a theatrical one: for *this specific action*, what exactly gets undone, by what mechanism, how long is the window to do it, and can you prove afterward that it happened? A vendor who can answer all four for one named action is describing a control. A vendor who answers “yes, we support rollback” is describing a hope.

Rung	What actually happens	Data-loss risk	Evidence it produces	Acceptable use
1 · Per-action undo	The platform reverses exactly the one action and restores the prior state of that object alone	None beyond the action reversed	A per-transaction record of the action and its matched reversal	Any step where a clean, attributable reversal is required
2 · Semantic compensation	A defined reverse business action offsets the original; the original event still stands (a refund against a charge)	None to unrelated data; the original record remains	Correlated forward and compensating entries	Multi-step work where state can’t be rewound but can be offset
3 · Point-in-time restore	The whole dataset is rewound to an earlier snapshot	High — every legitimate change since the snapshot is lost	A restore log, not a per-action trace	Catastrophic corruption or bulk loss only
4 · Manual remediation	A person fixes it by hand	Depends on the	Only what the operator	Last-resort recovery, not

Rung	What actually happens	Data-loss risk	Evidence it produces	Acceptable use
	after the fact	operator; error-prone	chooses to record	a routine control
5 · Support ticket	A human promises to fix it later	Unbounded and time-dependent	Usually none until it is resolved	Not acceptable for any consequential action

> **Grade per action, never per platform**
>
> The same product can be rung 1 for a draft edit and rung 5 for an outbound payment. Reversibility is a property of each action, not of the software as a whole, so never accept a single platform-wide grade — ask the question for each action your workflow will actually take.

2. Why “we restore from backup” is not rollback

This is the most common and most expensive conflation in the room, and it reaches signed contracts because both parties speak true sentences. The buyer hears rung 1 — a surgical undo of one action. The vendor means rung 3 — a whole-dataset rewind. They are not the same control, and only one answers the question a compliance reviewer is actually asking.

Dimension	Per-action reversal	Point-in-time restore
Granularity	One action, one object	The entire dataset as of a snapshot
Collateral impact	Nothing else touched	Every legitimate change since the snapshot is undone too
Concurrent changes	Preserved	Lost
Evidence produced	Action correlated to its reversal, per transaction	A restore event, with no per-action accountability
Speed	Immediate, targeted	Bounded by how long a full restore takes

Dimension	Per-action reversal	Point-in-time restore
Tenant / action isolation	Reverses one tenant's one action inside a shared system	Cannot isolate one action without collateral loss
Cost profile	Low, routine	High, disruptive — a recovery event
When it's the right tool	The routine "the automation did one wrong thing" case	Corruption, ransomware, or bulk data loss

Two numbers hide inside "we can restore." The **recovery-point objective** is how much recent data you accept losing (how far back the snapshot sits); the **recovery-time objective** is how long you are down while it runs. Ask both in writing — a restore capability quoted without them is a claim, not a commitment.

> **The audit trap**
>
> If the activity record is written inside the same transaction as the business change, then rolling that transaction back erases the evidence too — you lose the proof that the action ever happened. The record must commit on a separate path from the data it describes. A quick test: ask the vendor to reverse one action from last week without affecting anything else that happened that day, and watch whether they reach for a full restore.

3. How real reversibility is engineered

You do not need to build these mechanisms to evaluate them — but recognising them lets you ask questions a marketing deck cannot answer. Real reversibility rests on a few well-understood engineering ideas, described here in plain terms.

Undo built alongside the action, not after the incident

In a sound design, every reversible action is registered together with a defined, tested way to undo it — the unwind path exists before the action ever runs. A reversal path invented under pressure during an incident is not a tested reversal path; it is an improvisation you are discovering at the worst possible moment.

Compensation, not a naive rewind

A multi-step workflow is unwound by a sequence of compensating actions — each step carries its own defined reverse. This is a business-rule reversal, not a data rewind: a refund is not the deletion of a charge, and it may issue only a partial amount. Compensation may run steps in a different order than the original, may not restore the exact prior state, and must account for legitimate changes that happened in between. Be suspicious of any vendor who claims they "just replay it backwards" — that describes a rewind, which the domain rarely allows.

Safe retries

Automation retries when a network call times out. The safety rail is that the same operation applied twice must land exactly once — enforced at the data layer by a unique key plus an explicit "in progress" state, not by hopeful handling of a duplicate request. Without it, a retried undo can double-refund or double-cancel. Ask where the reversal's state lives: a real implementation durably records, per step, both what it did and how to undo it, so an interrupted unwind can resume from the point of failure rather than restart or stall.

Beneath the specific mechanisms sit six properties that a genuinely safe platform can demonstrate on a real action rather than describe in principle. Ask to see each one.

Undo by design

Every reversible action has a built-in, tested reversal, so there is always a way back.

Make them show it: pick one action and ask them to reverse it live.

Confirm before commit

Consequential or irreversible actions stop and wait for a named person before running.

Make them show it: ask which action classes are gated, and who can be the confirmer.

No self-approval

The system must be unable to fabricate its own sign-off; a real person has to act, so it cannot authorise itself.

Make them show it: ask how the platform proves an approval came from a person, not from the automation.

A record that can't be rewritten

Every action and every undo is written to a log no one — including the vendor — can edit or delete.

Make them show it: ask who *can* alter the log. The right answer is "no one."

Stops when unsure

If a permission, check, or version doesn't line up, it halts and escalates instead of guessing.

Make them show it: ask what happens on ambiguity. The safe default is "stop," not "proceed."

Least access per task

Each function runs with only the permissions it needs, so a fault or leak in one path can't reach another.

Make them show it: ask whether one compromised component exposes everything, or only its own slice.

> **Red flag: an undo with no thread back to the action**
>
> Watch for "undo" implemented as a second forward action with no linkage to the original — no shared correlation identifier, no recorded compensation, no way to prove the reversal matched the action it claims to reverse. An undo you cannot tie to its action is not evidence of anything.

4. The line you cannot cross: irreversible actions

Every workflow has points of no return. Sending money, filing a legally binding submission, sending an external email, deleting data held by a third party — no compensation fully unwinds these. A "reversal" here is a new business event, not an erasure. A vendor who tells you "everything is reversible" either misunderstands the domain or is quietly selling you the bottom of the ladder.

Two engineering disciplines make these actions safe to run at all. First, position an irreversible step *last* and run it only after every validation has passed, so a failure earlier in the workflow costs nothing that can't be taken back. Second, gate it behind a confirm-before-commit stop: the action does not run unattended; it halts and requires a named human to confirm, turning an un-undoable action into a deliberate, attributed decision. The platform must be structurally unable to fabricate that sign-off — a real human act is required — and where a recovery decision is high-impact or

genuinely ambiguous, the safe design pauses for a person rather than auto-guessing between “continue” and “unwind.”

Example action	Reversibility class	Required control
Edit an internal draft or record with a transaction behind it	Reversible	Automatic, tested per-action undo; may run unattended
Change a status others may act on downstream	Compensable	Defined compensating action + monitoring for a fast unwind
Grant an access entitlement	Compensable	Confirm gate or fast revoke, with the grant and revoke both recorded
Send an external email	Irreversible	Named-human confirm gate before send
Release a payment or transfer funds	Irreversible	Confirm gate; positioned last, after all validations
Delete data held by a third party	Irreversible	Confirm gate; no unattended path
File a regulatory or legally binding submission	Irreversible	Confirm gate; attributed human decision on record

Buyer move: ask the vendor to classify the actions in *your* intended workflow into these three classes, and to show you the confirm gate on each irreversible one.

5. Proving it can’t be faked: audit, fail-closed, least privilege

A reversibility claim is only as trustworthy as the properties that stop it from being quietly bypassed. Three structural traits turn “we can undo it” from an assertion into something you can evidence.

An independent, append-only record

Every action and every reversal is written to a trail that cannot be edited or deleted after the fact — write-once storage, in the general sense that alteration and deletion are prevented, not merely discouraged. Critically, that record must survive a rollback of the business data: it is written on a separate path with its own provenance — who acted, when, from where — so even a failed or reversed operation still leaves a

permanent trace. This is what lets a reversal be proven to have matched its action rather than asserted to have.

Fail-closed by default

When a check, a permission, or a version doesn't line up, the step stops and escalates rather than committing something it can't take back. "When unsure, don't" is the only safe default for an action with no undo. The dangerous alternative — quietly skipping a step or proceeding on a guess — hides the exact failure a reviewer needs to see.

Least privilege and separation of duties

Each function runs with only the credentials and scopes it needs, so a fault or leak in one workload cannot reach another — this bounds the blast radius any single rollback ever has to cover. Carry the separation-of-duties principle long applied to human financial controls into the automation itself: the identity that initiates an action cannot be the one that approves it, and the approver cannot later credibly deny it. Mature platforms enforce all of this at execution time — blocking the unsafe step — rather than discovering violations in an after-the-fact report.

> **Trustworthy-reversal evidence test**
>
> Ten things a reviewer should be able to tick against a live demonstration, not a document:
>
> - An immutable audit entry is produced for the action.
> - A second immutable entry is produced for the reversal.
> - The reversal is correlated to the exact action it reversed.
> - The audit record survives a rollback of the business data.
> - Each entry carries who acted, when, and from where.
> - A version or permission mismatch causes a visible stop, not a silent proceed.
> - An irreversible action halts for a named human confirmation.
> - The confirmation cannot be scripted or auto-filled by the automation.
> - The function acted under a scope limited to its own workload.
> - The whole trail can be exported in a tamper-evident, attributable form.

6. The vendor scorecard

Score on *evidence*, not on the answer alone. **2** = demonstrated on a live system; **1** = documented but not shown; **0** = asserted only. Weight the bands for your sector — raise the Evidence band for records-retention regimes; raise the Boundaries band for payments and public-facing filings. Confirm which regimes apply, and the exact obligations they impose, with your own counsel or authorising official.

Band & question	Weight	Score
Mechanism · Is undo built and tested alongside each reversible action, or bolted on later?	**High**	0 / 1 / 2
Mechanism · For a named reversible action, what exactly is restored, and by what mechanism?	**High**	0 / 1 / 2
Mechanism · Applied twice, does the same operation land exactly once?	**Med**	0 / 1 / 2
Boundaries · Which actions in our workflow are irreversible, named explicitly?	**High**	0 / 1 / 2
Boundaries · Does every irreversible action stop for a named human before it commits?	**High**†	0 / 1 / 2
Boundaries · Can the platform fabricate or auto-fill that human approval?	**High**†	0 / 1 / 2
Evidence · Is every action and reversal written to a record no one — including the vendor — can edit?	**High**†	0 / 1 / 2
Evidence · Is the record written separately, so a rollback of the data doesn't erase it?	**High**	0 / 1 / 2
Evidence · Can a tamper-evident, attributable trail be exported on demand for any action and its reversal?	**High**	0 / 1 / 2
Operations · Are the recovery-point, recovery-time, and per-action reversal windows stated in writing?	**Med**	0 / 1 / 2
Operations · On a permission, version, or precondition mismatch, does it stop or proceed?	**High**	0 / 1 / 2
Operations · Who is notified, how fast, when it fails closed or an irreversible action is confirmed?	**Med**	0 / 1 / 2

†Disqualifiers: a "yes" on self-fabricated approval, an editable audit record, or any irreversible action with no human confirm gate ends the evaluation regardless of the

total. Otherwise, sum the bands and compare — but treat any **0 on a High-weight row as a stop, not a deduction.**

> **Worked example**
>
> Vendor A demonstrates a live per-action undo, a confirm gate on every irreversible step, and a clean tamper-evident export — twos across the High rows. Vendor B posts a higher raw total on convenience features but can only "restore from backup" (rung 3) and concedes its admins can edit the log. Vendor B is out on the disqualifiers alone: a process that cannot reverse one action cleanly, and cannot produce an untouchable record of what happened, is one you cannot defend in an investigation, whatever else it does well.

7. Run the fire drill: a hands-on acceptance test

Claims are cheap; behaviour is not. Reserve final judgement until you have watched the platform undo a real action in a sandbox that mirrors your workflow. Script the five tests below, run them in a trial or pilot, and record each result against the scorecard.

Test	What you do	Expected safe behaviour	Pass?
1 · Surgical reversal	Perform one action, then reverse only it	Nothing else in the window changes; an immutable entry records both the action and its reversal	☐
2 · The irreversible gate	Attempt a consequential/irreversible action unattended	It stops and demands a named human confirmation you cannot script around	☐
3 · Fail-closed	Introduce a version, permission, or precondition mismatch mid-workflow	The step halts and escalates rather than committing a partial, un-undoable change	☐
4 · Retry safety	Fire the same operation twice, simulating a network retry	It lands exactly once; the duplicate is	☐

Test	What you do	Expected safe behaviour	Pass?
		recognised, not reprocessed	
5 · Evidence export	Export the audit trail for the test	It is tamper-evident, attributable, and complete enough to hand an auditor	☐

A platform that passes on a live pilot is worth more than ten that pass on a slide. If you cannot export clean evidence, you do not have a control — you have a demonstration.

8. Lock it into the contract and operations

Reversibility that lives only in the sales cycle decays the moment the platform changes. These are the properties worth converting into enforceable, monitorable commitments — and worth confirming with your own counsel before signing.

Clause to require	The failure it prevents
Recovery-point and recovery-time objectives, in writing	A restore capability whose real data-loss and downtime cost is discovered during an incident
Per-action reversal window, stated as a term	An undo that silently expires before you knew there was a clock
Audit-retention period, stated as a term	Evidence deleted before an investigation or records obligation needs it
A maintained register of which actions are reversible, compensable, or irreversible	A later feature quietly downgrading an action's rung without notice
Evidence-on-demand: a tamper-evident export for any action and its reversal within a defined time	Standing audit-readiness collapsing into a scramble when a regulator asks
Notification and escalation terms for fail-closed stops and confirmed irreversible actions	A safe stop turning into a silent outage because no one was told
Change-control notice and re-testing rights on the safety machinery	Paired undo, confirm gates, immutable audit, or scoping altered without your knowledge

Clause to require	The failure it prevents
Named sub-processors and model providers, kept current	Inherited risk from dependencies carved out of the vendor's own scope
A right to re-run this acceptance test periodically against production-like config	A control that worked at signing and drifted since — indistinguishable from one that never existed

How to run the evaluation

Grade rollback per action, never per platform. Send the scorecard ahead of a working session so answers arrive with evidence attached. In the session, make the vendor *demonstrate* the reversal, the confirm gate, the fail-closed stop, and a clean audit export — then run the five-test fire drill in a pilot before you commit. Route the recovery, retention, and change-control terms through counsel, and schedule the fire drill to repeat against production-like config, because a reversibility control verified once is not a control verified forever. Nothing here is legal advice; it is the list of things to have your own advisers confirm.

SG1 CONSULTING

sg1consulting.us · contact@sg1consulting.us