Testing change management when AI writes or approves code

Nov 12, 2026 · @Christian

Status and purpose

Draft v0.1 for review by practising IT auditors. It proposes how to test a change-management control when AI assistants write code, or bots and agents approve or merge it.

- **For:** external and internal IT auditors testing SOC 2, ISO 27001, SOX ITGC or PCI DSS change controls, and the teams preparing for them.
- **Vendor-neutral:** every test can be run with git, the code host's audit log and standard tools. No product is needed.
- **Not:** an audit standard, a certification scheme or an opinion on any control. The auditor's judgement stays with the auditor.
- **Drafted by** Christian Onwe (Toevan OÜ, Tallinn). Reviewers who contribute are credited by name if they agree.

Why the usual tests break

A standard change test asks: was the change authorised, reviewed by someone other than the author, tested, and approved before it reached production? Each question assumed a human author and a human approver. AI changes four things.

What the test assumes	What can now happen
The author is a person accountable for the change	An assistant or agent wrote most or all of it; the commit names a person or a bot account
The reviewer is independent of the author	The same agent, or one configured by the author, writes the change and approves it
An approval means a person looked at the code	A bot account or auto-merge rule approves or merges without a human
The population is the PRs a team opens	Agents open and merge many small changes, some outside the usual PR flow

Practitioners we spoke to in November 2026 say auditors are still learning how to test this. One put it as: "If the auditor doesn't know how to test AI, the whole testing falls apart." Auditors may also be unsure what to test, and whether any required data is missing from the test scope.

Scope and framework mapping

In scope: changes to code that reaches production through a git host (GitHub first), for the repositories the audit scoping lists. Out of scope: infrastructure changes made outside git, the AI tools' own security, and agents' credentials and permissions (tested under logical access controls).

Framework	Requirement the tests support	Tests
SOC 2 (2017 TSC)	CC8.1 change management	T1–T6
ISO/IEC 27001:2022	Annex A 8.32 change management; 8.25 secure development life cycle	T1–T6
SOX ITGC	Program change management	T1–T6
PCI DSS v4.0	Requirement 6.5.1 changes to system components	T1–T5

Clause references are for reviewers to check against the version their client is audited to.

Terms

- **Assistant:** suggests code to a person, who writes the commit (editor autocomplete, chat).
- **Agent:** writes code and opens or updates changes itself, with or without a person prompting it.
- **Bot or app account:** a non-human identity on the code host, such as a GitHub App or a service account.
- **Auto-merge:** a rule that merges a change once conditions are met, with no person pressing merge.

Test procedure

Seven steps, run per in-scope repository for the audit period. T0 sets the risk tiers that the later tests use. T1 and T2 come next, because every later test depends on knowing the full population and who each actor is.

T0. Repositories are tiered by risk

- Request: the team's classification of in-scope repositories, for example: affects financial reporting, payments or customer data; internal tooling; UI or content only.
- Test: agree the tiers to what each repository does. Use them to set where human review is mandatory (T4) and to weight the sample: more items from high-risk repositories and from agent-authored changes, since agents produce many small changes that dilute a random sample.
- Red flags: no tiering; a payments or ledger repository classed as low risk.

T1. Population includes every change, however it was made

- Request: the list of changes the team provides as the population.
- Test: recount from git. List every commit that reached the default branch in the period and match each to a pull request. Investigate commits with no PR (direct pushes, bot pushes, force-pushes).
- Red flags: population drawn only from a deployment tool or PR list; bot or agent commits missing from it.

T2. Every actor is identified as a person, a bot or an agent

- Request: the list of bot accounts, apps and AI agents with write, approve or merge rights, and the person accountable for each.
- Test: for each sampled change, classify author, approver and merger. Agree bot and app accounts to the code host's installed apps and tokens.
- Red flags: agents acting through a person's own account; accounts with merge rights that nobody owns.

T3. The approver is independent of the author

- Test: the approver is not the author, and is not an agent the author configured or prompted. The approval applies to the final commit, not an earlier one.
- Red flags: the same agent writes and approves; approvals dismissed or stale after new commits.

T4. A person reviewed the change where policy requires one

- Request: the policy stating which changes need human review (for example, money-moving code but not UI copy).
- Test: for those changes, at least one approval came from a person. Check auto-merge rules and bot approvals against the policy.
- Red flags: auto-merge enabled on critical repositories; policy silent on AI or bot approval.

T5. The controls themselves stayed on

- Request: the code host's audit log for the period.
- Test: list every change to branch protection, rulesets and required checks, every admin bypass and every switch-off. Match changes merged during those windows.
- Red flags: review rules switched off and on again inside the period with no ticket.

T6. AI involvement is declared where policy requires it

- Request: the policy on declaring AI assistance, if there is one.
- Test: check the declaration is recorded for sampled changes (PR template field, commit trailer, label). Treat it as declared, not verified: no tool can reliably tell AI-written code from human-written code.
- Red flags: a report claiming to detect AI authorship; no policy at all for repositories in scope.

Known limit, by design. AI authorship cannot be verified, and this procedure does not depend on it. T1 to T5 hold whoever or whatever wrote the code: every change is in the population, every actor is classed, approvals are independent and on the final commit, and the controls stayed on. T6 only tests whether the team's own declaration policy operates.

Request list

What to ask the client for, ready to copy into a request list.

Item	Used in
Risk classification of in-scope repositories	T0, T4
Read access to the repositories, or an export of commits on the default branch for the period	T1
The population of changes the team uses, and how it was produced	T1
Bots, apps and AI agents with write, approve or merge rights, and the owner of each	T2
Branch protection and ruleset settings at the start of the period	T3, T4
Policy on which changes need human review, including bot and AI approvals	T4
Code host audit log for the period	T5
Policy on declaring AI assistance, if any	T6

Evidence quality

For each fact used in a test, the auditor should know how it was obtained, because that decides how far to rely on it.

How the fact was obtained	Example	Reliance
Recomputed by the auditor from git	Commit on the default branch; author equals approver	Highest: re-performed
Reported by the code host, captured at the time	Approval on the final commit; branch rule switched off at 14:02	High if captured when it happened and protected from later edits
Produced by the team (screenshot, export, pre-generated report)	PR approval screenshot; CSV of merged changes	Only after testing completeness and accuracy: it can be edited, cropped or produced outside the audit window. Prefer observing it being generated, or recompute it from the source
Declared by the team	"This change was AI-assisted"	Shows the policy operated; not proof of authorship
Not available	Reason for a direct push, never given	Record as not captured; never infer

Logs and reports are only as reliable as their protection from change. Append-only or signed records, or records held by a third party such as the code host, need less testing than files the team holds and can edit.

Open question: the audit window. Auditors often require evidence to be produced inside the audit window, so an export taken months earlier is not accepted. Records captured when each change happened are older than the window by design. Proposed answer: the auditor re-verifies them during the window (recount, signature check, cross-check against the code host's audit log), and that re-verification is the in-window evidence. Reviewers: would this satisfy you?

Questions for reviewers

Comments on any line are welcome. These five matter most:

1. Would you run T0–T5 as written? Which would you drop, and what is missing? Is T6 worth keeping as a policy test?
2. Is an approval by a bot or agent ever acceptable evidence of review in your testing? Under what conditions?
3. When the author and approver are both agents set up by one engineer, do you treat that as a segregation-of-duties failure?
4. Would re-verifying records inside the audit window satisfy your timing requirement (see Evidence quality)?
5. Have clients or your firm started asking about AI in change management? Since when?

Reply by comment or to christian.onwe@toevan.com.