

# Copilot-Authored Commits: Agent Change Attribution

## Problem

When the Copilot agent edits files via tools ( `create_file` , `replace_string_in_file` , `apply_patch` , etc.), those changes are written directly to disk with no distinction from developer-made edits. This creates several issues:

- **No attribution** — `git log` shows the developer as the author of AI-generated code
- **No auditability** — impossible to tell which changes were human vs. machine after the fact
- **No traceability** — no record of which model or tools produced the code
- **Compliance risk** — organizations increasingly require disclosure of AI-generated code in version control

GitHub Copilot's own coding agent solves this by committing as `copilot[bot]` , but that's a server-side bot account unavailable to third-party integrations.

## Constraints

- **No official Copilot email** — GitHub's `copilot[bot]` is server-side only; no public email exists for use in local `git commit --author`
- Creating a dedicated GitHub machine user costs a seat and requires org admin setup
- GitHub Apps can author commits via API but not through local `git commit`
- Git's `--author` flag requires `Name <email>` format
- IntelliJ's `CheckinHandler.beforeCheckin()` can only return `COMMIT` or `CANCEL` — it cannot modify the commit author or message

## Solution

Two git-native mechanisms that require no GitHub account setup and are parseable by CI/tooling:

### 1. `--author` flag — separates who wrote vs. who approved

Git natively separates **author** (who wrote the code) from **committer** (who approved it). We set the author to Copilot, leaving the committer as the developer.

For the email, we use `copilot@copilot.example` — a reserved domain under RFC 6761 guaranteed to never be a real address.

```
git commit --author="GitHub Copilot (gpt-4.1) <copilot@copilot.example>" -m "Add auth endpoint"
```

Author: GitHub Copilot (gpt-4.1) <copilot@copilot.example>  
Commit: John Doan <john.doan@citi.com>

This gives immediate separation — `git log --author="GitHub Copilot"` returns only AI-generated commits, while the committer field preserves accountability.

## 2. Generated-by trailer — structured metadata for tooling

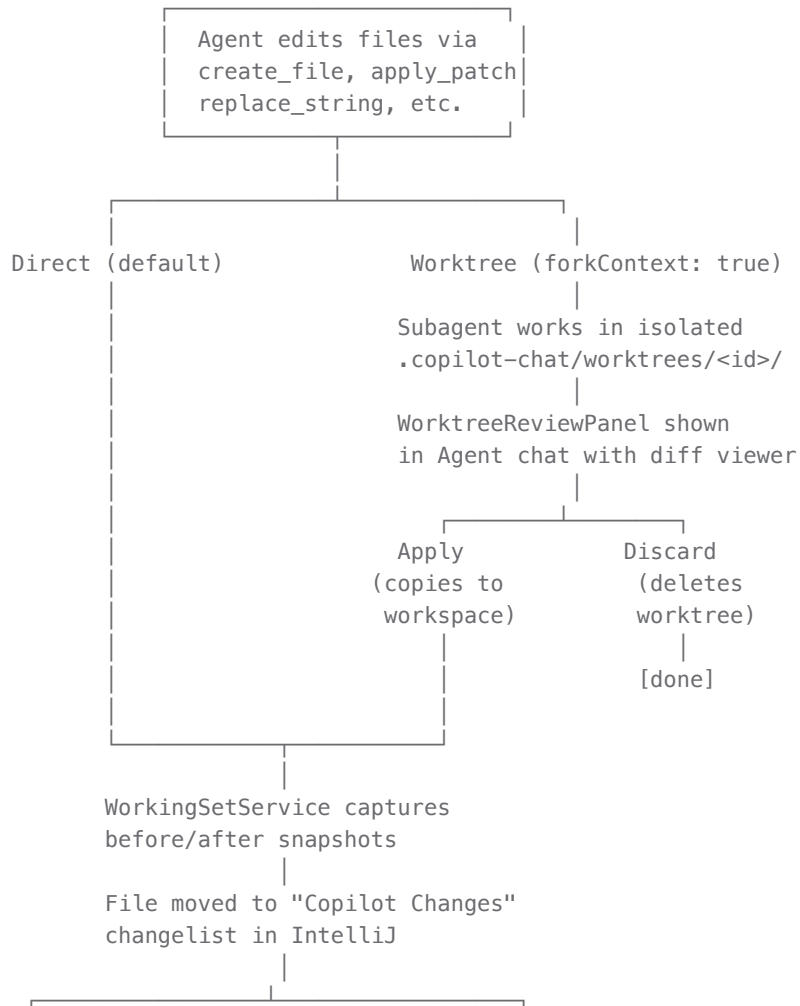
Git trailers are key-value metadata at the end of a commit message. GitHub renders them in the commit detail view, and CI pipelines can parse them.

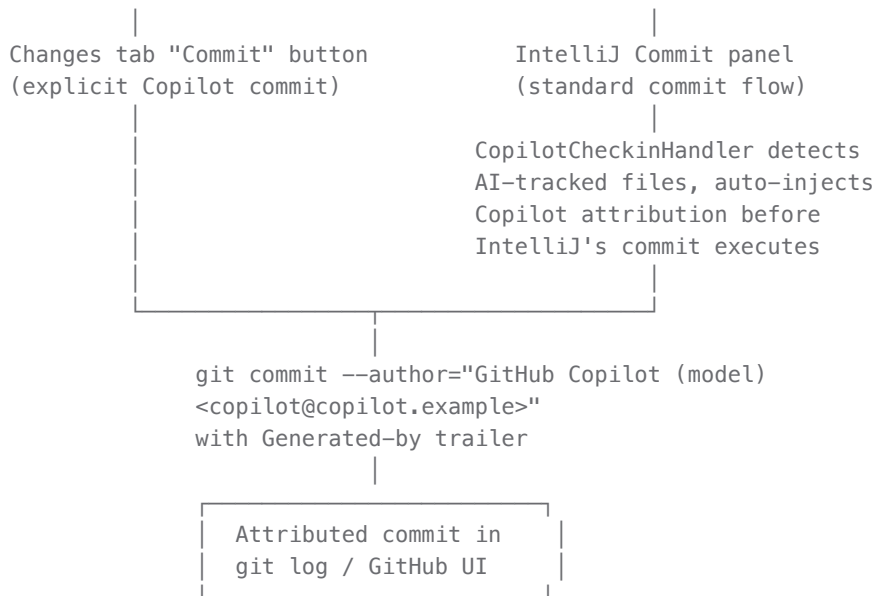
Add auth endpoint

Generated-by: github-copilot

## End-to-end flow

There are three commit paths. All agent paths produce the same attributed commit format.





## Auto-inject attribution

The plugin enforces correct attribution regardless of which commit path the developer uses. There is no "commit anyway under your name" escape hatch.

### How it works

`CopilotCheckinHandlerFactory` hooks into IntelliJ's standard commit flow via the `CheckinHandler` API. When the developer clicks Commit in any commit panel:

1. **Detect** — `beforeCheckin()` checks if any files in the commit are tracked by `WorkingSetService`
2. **Separate** — AI-tracked files are extracted from the commit set
3. **Commit** — `CopilotCommitService` writes the agent's known content to disk, runs `git commit --author="GitHub Copilot (model) <copilot@copilot.example>" -- <paths>` with the `Generated-by` trailer, then restores the original disk content
4. **Notify** — A balloon notification reports the result: "Committed N file(s) as GitHub Copilot"
5. **Continue** — If non-AI files remain, IntelliJ's normal commit proceeds for those files under the developer's name

This write-stage-restore approach works around the `CheckinHandler` API limitation (can only return COMMIT/CANCEL, cannot modify author or message) by executing a separate `git commit` for the AI files before IntelliJ's commit runs.

### Return value logic

Scenario	Handler returns	Effect
No AI files in commit	COMMIT	Normal IntelliJ commit
AI files committed, non-AI files	COMMIT	AI files committed as Copilot; IntelliJ commits the rest under

Scenario	Handler returns	Effect
remain		developer's name
AI files committed, no remaining files	CANCEL	AI files committed as Copilot; commit dialog closes
Attribution commit fails	COMMIT	Graceful fallback — all files commit under developer's name with warning

## Hunk-level diff tracking

`HunkDiffEngine` uses IntelliJ's `ComparisonManager.compareLines()` API to compute which line ranges were modified by the agent. This enables:

- Logging which specific hunks are agent-authored in each commit
- Future per-line attribution for mixed files (files with both AI and developer edits)

For new files, the entire file is treated as a single agent hunk. For modified files, only the changed line ranges are attributed to the agent.

## Worktree isolation for subagents

When a subagent has `forkContext: true` in its `.agent.md` definition, it operates in a git worktree rather than the main workspace. This is primarily a **parallel agent safety** mechanism — multiple subagents can't overwrite each other's changes — but it also adds an approval gate before changes enter the attribution pipeline.

The subagent works in `.copilot-chat/worktrees/<agentId>/` on branch `copilot-worktree-<agentId>`. Its tool calls are routed to the worktree via a per-conversation workspace override in `ToolRouter`. When the subagent completes, a diff is generated against the main workspace and a review panel appears inline in the Agent chat. The user can:

- **Apply** — files are copied to the main workspace and registered with `WorkingSetService`, entering the normal attribution pipeline (Copilot Changes changelist → auto-attributed commit)
- **Discard** — the worktree is deleted with no effect on the main workspace

If no files changed, the worktree is cleaned up automatically. Stale worktrees from crashes are pruned on startup.

## Change source summary

Change source	Approval gate	Commit author	Trailer
Developer edits	None (normal flow)	Developer	None
Agent tools (direct)	Changes tab review	GitHub Copilot (model)	Generated-by

Change source	Approval gate	Commit author	Trailer
Agent tools (worktree)	Worktree review → Changes tab	GitHub Copilot (model)	Generated-by

Both agent paths produce identical commits. The worktree path adds a pre-review step before changes reach the Working Set. Attribution is enforced automatically on all commit paths — the developer does not need to use a specific button or workflow.

## Querying AI-generated commits

```
# All Copilot commits
git log --author="GitHub Copilot"

# All commits with the Generated-by trailer
git log --grep="Generated-by: github-copilot"

# Count of Copilot vs developer commits
echo "Copilot: $(git log --author='GitHub Copilot' --oneline | wc -l)"
echo "Developer: $(git log --author='GitHub Copilot' --invert-grep --oneline | wc -l)"
```

On GitHub, the `--author` field shows in the commit detail view and PR commit list. The `Generated-by` trailer renders as structured metadata below the commit message. PR reviewers can see at a glance which commits were AI-authored.

## Comparison with alternatives

Approach	Attribution in git log	GitHub UI	Requires account	Parseable by CI
<code>--author</code> + <b>trailer</b> (ours)	Author field + trailer	Trailer in commit view	No	Yes
Co-authored-by trailer	Trailer only	Co-author avatar	No	Yes
GitHub machine user	Real author with avatar	Full profile link	Yes (1 seat)	Yes
GitHub App	<code>app[bot]</code> author	Bot badge	Yes (app setup)	Yes
Commit message prefix <code>[copilot]</code>	Message only	Visible but unstructured	No	Fragile regex

## Industry precedent

- **Claude Code** appends `Co-Authored-By: Claude <noreply@anthropic.com>` to commits

- **GitHub Copilot coding agent** commits as `copilot[bot]` (server-side only)
- **Cursor** relies on manual attribution via `.cursorrules` configuration
- **SSW Rules** recommends co-author trailers for all AI-assisted commits

The `--author` + trailer approach provides stronger attribution than co-author trailers (Copilot is the *author*, not a co-author) while remaining fully local with no account dependencies.