April 13, 2026 · 7 min read · By Yiğit Konur

auto-approve Claude Code plan mode via PermissionRequest

ai software-engineering

if you're using Claude Code's plan mode, you've probably seen the "ready to code?" approval dialog. as I always approve Opus 4.6's plan, no need to do this thing manually anymore. turns out you can automate that click with a single hook once you target the right event.

this also highlights a tiny docs gap tracked in anthropic's repo: issue #11891 (https://github.com/anthropics/claude-code/issues/11891) ("[DOCS] Missing PermissionRequest hook details in Hooks Guide and Input Reference schema"). + not well-documented and we need to do better job on documenting JSON payload details as Claude Code is not open-source, we had to do basic reverse engineering (not cool as its name, basically, just add event listener that save JSONs into a folder, and read from there, not assembly patched xD)

the key detail: it's PermissionRequest (not Stop)

the approval dialog is a `PermissionRequest` for the tool `ExitPlanMode`.

`Stop` fires after a run is done. plan approval happens while Claude is *waiting* for user input, so `Stop` is the wrong place to intercept it.

what PermissionRequest sends (stdin)

here's the (currently under-documented) shape you get on stdin when the dialog pops:

-

```json
{
  "session_id": "abc123-def4-5678-ghij-klmnopqrstuv",
  "transcript_path": "/Users/you/.claude/projects/-Users-you-my-project/abc123.jsonl",
  "cwd": "/Users/you/my-project",
  "permission_mode": "plan",
  "hook_event_name": "PermissionRequest",
  "tool_name": "ExitPlanMode",
  "tool_input": {
    "allowedPrompts": [
      { "tool": "Bash", "prompt": "run tests" },
      { "tool": "Bash", "prompt": "install deps" }
    ],
    "plan": "# Plan: Implement Feature X\n\n## Context\n\n..."
  }
}
```

two useful bits:

- `tool_name` tells you *what* is asking (here: `ExitPlanMode`)
- `tool_input.plan` contains the full plan markdown (which you can archive elsewhere)

minimal auto-approve hook (copy/paste)

script: `~/.claude/hooks/auto-approve-plan.sh`

```bash
#!/bin/bash
cat >/dev/null  # consume stdin (important on Windows/WSL)
cat <<'EOF'
{"hookSpecificOutput":{"hookEventName":"PermissionRequest","decision":{"behavior":"all
EOF
```

config (only match `ExitPlanMode`, not everything):

-

```json
{
  "hooks": {
    "PermissionRequest": [
      {
        "matcher": "ExitPlanMode",
        "hooks": [
          {
            "type": "command",
            "command": "~/.claude/hooks/auto-approve-plan.sh"
          }
        ]
      }
    ]
  }
}
```

notes that save time:

- your hook should read stdin (otherwise you can hang)
- your output must be wrapped under `hookSpecificOutput` with `hookEventName: "PermissionRequest"`

optional: archive each plan to Craft.do

because the plan markdown is already in `tool_input.plan`, you can ship it to Craft.do in the background and still return the approval immediately.

the only “gotcha”: don’t pull the markdown into a shell variable and then re-embed it into JSON (multiline content will bite you). build the Craft.do payload in a single `jq` pass instead.

```bash
#!/bin/bash
TMPFILE="$(mktemp)"
cat > "$TMPFILE"
```

-

fire-and-forget archive (don't block the approval response)

```bash
(
jq
--arg ts "$(date '+%Y-%m-%d %H:%M')"
--arg pageId "$CRAFT_PAGE_ID"
--arg home "$HOME"
'{
blocks: [{
type: "page",
textStyle: "card",
markdown: ("[" + (.cwd | sub($home; "~")) + "] - [" + $ts + "]"),
content: [{ type: "text", markdown: .tool_input.plan }]
}],
position: { position: "end", pageId: $pageId }
}' < "$TMPFILE"
| curl -sS -X POST "$CRAFT_API_URL/blocks" -H "Content-Type: application/json" -d @-
>/dev/null 2>&1
) &

rm -f "$TMPFILE"
```

approve the dialog

```bash
cat <<'EOF'
{"hookSpecificOutput":{"hookEventName":"PermissionRequest","decision":{"behavior":"all
EOF
```

if you want this packaged: yigitkonur/hooks-claude-approve

i bundled the "auto-approve + (optional) Craft.do archive" flow into yigitkonur/hooks-claude-approve (https://github.com/yigitkonur/hooks-claude-approve).

-

modes:

install:

```bash
bash <(curl -fsSL https://raw.githubusercontent.com/yigitkonur/hooks-claude-approve/ma
```

links

- hooks tool: https://github.com/yigitkonur/hooks-claude-approve (https://github.com/yigitkonur/hooks-claude-approve)
- docs gap tracker: https://github.com/anthropics/claude-code/issues/11891 (https://github.com/anthropics/claude-code/issues/11891)
- official hooks guide: https://docs.anthropic.com/en/docs/claude-code/hooks (https://docs.anthropic.com/en/docs/claude-code/hooks)

SOURCES

Anthropic hooks guide (https://docs.anthropic.com/en/docs/claude-code/hooks)

Docs gap issue #11891 (https://github.com/anthropics/claude-code/issues/11891)

hooks-claude-approve (https://github.com/yigitkonur/hooks-claude-approve)