

Hookイベント一覧

| イベント               | 発火タイミング                       | 主な用途            | 制御効果                            |
|--------------------|-------------------------------|-----------------|---------------------------------|
| SessionStart       | セッション開始・resume・clear・compact後 | 開発文脈の読み込み、env注入 | ブロック不可                          |
| UserPromptSubmit   | ユーザー送信直後、処理開始前                | プロンプト検証、文脈注入    | exit2でブロック・消去                   |
| PreToolUse         | ツール実行の直前                      | 危険コマンド遮断、機密保護   | exit2 / permissionDecisionでdeny |
| PermissionRequest  | 許可ダイアログ表示時                    | 許可の自動承認・拒否      | JSON decisionで制御可               |
| PostToolUse        | ツール成功後                        | 整形、検証、ログ記録      | ブロック不可、exit2でClaudeへ指摘          |
| PostToolUseFailure | ツール失敗後                        | エラー記録、リトライ判断    | ブロック不可                          |
| Notification       | 通知発出時（許可待ち等）                  | デスクトップ通知の発火     | ブロック不可                          |
| Stop               | Claudeが応答を終えた時                | 完了通知、検証の起動、続行強制 | exit2で停止を阻止・続行                  |
| SubagentStop       | サブエージェント終了時                   | サブエージェントの続行制御   | exit2で停止を阻止                     |
| PreCompact         | コンテキスト圧縮の直前                   | 圧縮前の保全、圧縮の阻止    | exit2でブロック                      |
| SessionEnd         | セッション終了時                      | 後片付け、ログ記録       | ブロック不可                          |

settings.json 最小例

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "${CLAUDE_PROJECT_DIR}/.claude/hooks/guard.sh",
            "timeout": 30
          }
        ]
      }
    ]
  }
}
```

matcher記法

**"\*" / "/" / 省略** 全件にマッチ。毎回発火する

**"Bash"** 英数字・アンダースコア・|のみなら完全一致または区切りのOR  
Edit|Write

**正規表現** 上記以外の記号を含むとJavaScript正規表現として評価  
^Notebook / mcp\_memory\_.\*

**ツール系以外のmatcher対象** SessionStartはsource、Notificationは通知種別を取る  
startup|resume|clear|compact / permission\_prompt

出力制御（exit code / JSON）

**exit 0** 成功。stdoutをJSONとしてパース。JSONでなければ無音

**exit 2** ブロック。stdoutは無視しstderrをClaudeへ提示。効くのはPreToolUse / UserPromptSubmit / Stop / SubagentStop / PreCompact。Post系は実行済みのため指摘のみ

**その他の非0** 非ブロックエラー。transcriptにstderr先頭行が出て処理は続行

**"continue": false** Claude全体を停止。stopReasonにユーザー向け理由を添える

**"decision": "block" + "reason"** UserPromptSubmit / Post系 / Stop / SubagentStopでアクションを止める

**permissionDecision** PreToolUseのhookSpecificOutputで allow / deny / ask / defer

**additionalContext** UserPromptSubmit / SessionStartでClaudeへ追加文脈を渡す

**JSON制御の全体像** 下記の形でhookSpecificOutputに各イベント固有フィールドを入れる

```
{
  "continue": true,
  "stopReason": "停止理由 (continueがfalseのときユーザーへ)",
  "suppressOutput": false,
  "systemMessage": "ユーザーへの警告文",
  "decision": "block",
  "reason": "ブロック理由",
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason": "理由"
  }
}
```

ユースケース例

**PreToolUse: 危険コマンド遮断** matcher "Bash". rm -rf等を検出しdenyを返す

```
#!/bin/bash
CMD=$(jq -r ".tool_input.command")
if [[ "$CMD" == "rm -rf"* ]]; then
  echo '{"hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason": "破壊的削除を遮断"}}'
fi
exit 0
```

**PostToolUse: 整形/検証** matcher "Edit|Write". 編集ファイルにprettierをかける

```
jq -r ".tool_input.file_path" | xargs npx prettier --write
```

**UserPromptSubmit: 文脈注入** stdoutが追加文脈としてClaudeへ渡る

```
#!/bin/bash
echo "今日は $(date +%F)。社内規定 v3 を参照すること。"
# 終了コード0でstdoutがadditionalContextとしてClaudeへ渡る
exit 0
```

**Stop: 完了通知** 応答完了時にデスクトップ通知を出す

```
osascript -e 'display notification "Claudeが応答を完了しました" with title "Claude Code"'
```

**SessionStart: 環境注入** ブランチ・バージョンを流し込み、CLAUDE\_ENV\_FILEでenvを永続化

```
#!/bin/bash
BRANCH=$(git rev-parse --abbrev-ref HEAD)
echo "現在ブランチ: $BRANCH"
echo "node: $(node -v)"
# CLAUDE_ENV_FILEへ書けば以降のBashにenvが永続化される
echo 'export NODE_ENV=development' >> "$CLAUDE_ENV_FILE"
```

注意・Tips

**無限ループ防止** Stop / SubagentStopのhook内からclaudeを再呼び出ししない

**\$CLAUDE\_PROJECT\_DIR** プロジェクトルートの絶対パス。hookのenvにも設定され、commandやargsで \${CLAUDE\_PROJECT\_DIR} として展開できる

**stdout と stderr の差** stdout=Claudeへの入力（JSON or 文脈）、stderr=ユーザー表示。混同しない

**timeout既定値** command/http/mcp\_tool=600秒、UserPromptSubmit=30秒、prompt=30秒、agent=60秒。timeoutで上書き

**stdin JSON** session\_id / cwd / tool\_name / tool\_input / hook\_event\_name 等が渡る。jqで読む

**登録場所** ~/.claude/settings.json（全体）、.claude/settings.json（プロジェクト共有）、.claude/settings.local.json（共有しない）

**任意コード実行の自覚** hookは自分の権限でshellを実行する。信頼できないコマンドは登録しない

**引数のクォート** shell経由のためパスや変数は必ずクォートする。argsを使うexec形式なら特殊文字をそのまま渡せる